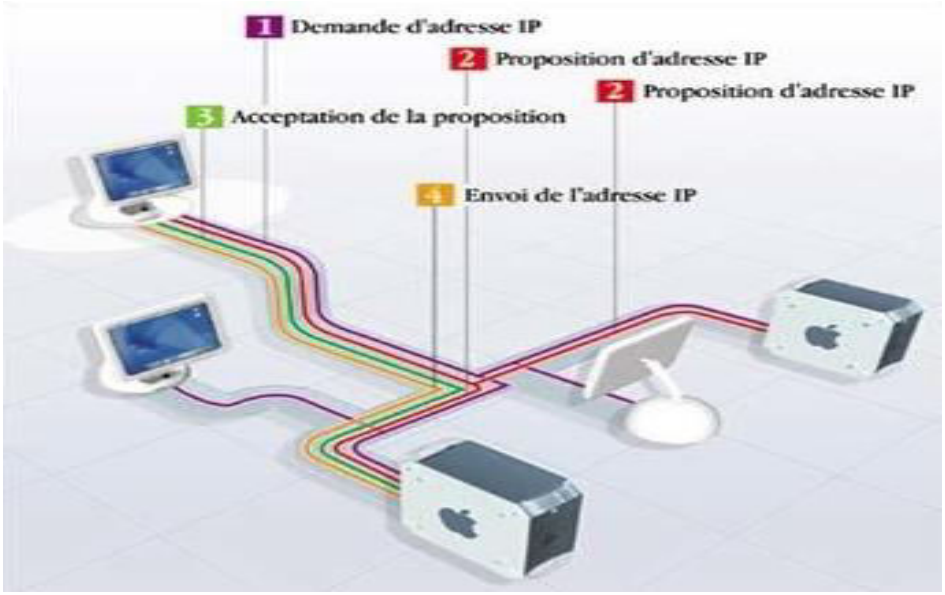
OFPPT

Office de la Formation Professionnelle
et de la Promotion du Travail

Administration Réseau sous linux


1 Demande d'adresse IP
2 Proposition d'adresse IP
2 Proposition d'adresse IP
3 Acceptation de la proposition
4 Envoi de l'adresse IP

Mehdi El Haouate

http://www.ofppt-tsri.blogspot.com/

https://www.facebook.com/TSRI.reseaux

OFPPT

Sommaire

- Configuration d'un réseau local (LAN).
  - Configuration d'une station
  - Serveur DHCP
  - Serveur DNS
- Partage de fichier
  - Service NFS
  - Service Samba
- Applications FTP, Telnet, SFTP et SSH.
- Routage, firewall.
- LDAP

OFPPT

Première partie

Configuration d'un réseau local (*LAN*)

Configuration d'une station

- **Configurer les interfaces à la main avec *ifconfig***

 # ifconfig

La commande **ifconfig** permet la configuration locale ou à distance des interfaces réseau de tous types d'équipements. Sans paramètres cette commande permet d'afficher les paramètres réseaux des interfaces.

La ligne de commande est:

```
#ifconfig interface adresse [paramètres]
```

Exemple:

```
#ifconfig eth0 192.168.1.2
```

Affecte l'adresse 192.168.1.2 à la première interface physique, ici eth0.

Configuration d'une station

- **Configurer les interfaces à la main avec *ifconfig***

Voici les principaux arguments utilisés:

interface: logique ou physique, il est obligatoire.

up: active l'interface

down: désactive l'interface

mtu: définit l'unité de transfert de paquet

netmask: affecter un masque de sous réseau

broadcast: définit l'dresse de broadcast.

Arp ou ***–arp*** activer ou désactiver l'utilisation du cache arp de l'interface

Metric, **multicast**…

Configuration d'une station

- **Configurer les interfaces à la main avec *ifconfig***

ifconfig peut être manipulée de différentes façons:

1- Afficher les paramètre réseaux de votre machine

```
# ifconfig
eth0 Link encap:Ethernet HWaddr 00:B2:3A:24:F3:C4
inet addr:192.168.0.2 Bcast:192.168.0.255 Mask:255.255.255.0
inet6 addr: fe80::2c0:9fff:fef9:95b0/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:6 errors:0 dropped:0 overruns:0 frame:0
TX packets:16 errors:0 dropped:0 overruns:0 carrier:5
collisions:0 txqueuelen:1000
RX bytes:1520 (1.4 KiB) TX bytes:2024 (1.9 KiB)
Interrupt:10

lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:16436 Metric:1
RX packets:92 errors:0 dropped:0 overruns:0 frame:0
TX packets:92 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:6040 (5.8 KiB) TX bytes:6040 (5.8 KiB)
```

Configuration d'une station

- **Configurer les interfaces à la main avec *ifconfig***

2- Désactiver une interface:

```
# ifconfig eth0 down
# ifconfig lo down
```

Si on lance un ping ou telnet sur localhost ou sur l'adresse de l'interface ethernet etho, aucune commande ne fonctionne, car même si la configuration IP est correcte, les interfaces sont désactivées.

3- On active ces interfaces par:

```
# ifconfig eth0 up
# ifconfig lo up
```

Configuration d'une station

- **Commande hostname**

Comme dans tout système, l'ordinateur sur lequel vous travaillez porte un nom. Sous Linux, ce nom est défini par le **hostname**. Il existe une commande « **hostname** » qui permet de définir le nom de la machine ou de visualiser ce nom.

Pour obtenir le nom de votre machine tapez hostname sans paramètres.

Pour changer la nom de votre machine tapez:

```
# sudo hostname nom_machine
```

Vous pouvez aussi changer le nom de votre machine d'une manière permanente en modifiant les deux fichiers /etc/hosts et /etc/hostname

Configuration d'une station

- **La commande arp**

La commande **arp** permet de visualiser ou modifier la table du cache ARP de l'interface. Cette table peut être statique et (ou) dynamique. Elle donne la correspondance entre une adresse IP et une adresse **MAC** (Ethernet)

A chaque nouvelle requête, la cache ARP de l'interface est mis à jour. Il y'a un nouvel enregistrement avec une durée de vie TTL (TimeToLive).

Voici un exemple de cache ARP obtenu avec la commande **`arp -va`**

```
farid@Serveur:/etc$ arp -va
? (192.168.226.2) at 00:50:56:e5:08:c4 [ether] on eth0
? (192.168.226.254) at 00:50:56:e4:a2:03 [ether] on eth0
Entries: 2   Skipped: 0   Found: 2
```

Configuration d'une station

- **La commande route**

Le routage définit le chemin .emprunté par les paquets entre son point de départ et son point d'arrivée. La commande **route** permet la configuration du routage sur un PC, switch ou routeur.

Voici un exemple de table de routage, obtenu avec la commande `route -n`

```
farid@Serveur:/etc$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.226.0   0.0.0.0         255.255.255.0   U     1      0        0 eth0
169.254.0.0     0.0.0.0         255.255.0.0     U     1000   0        0 eth0
0.0.0.0         192.168.226.2   0.0.0.0         UG    0      0        0 eth0
```

Syntaxe générale de la commande route:

route add [net | host] addr [gw passerelle] [métric coût] [netmask mask] [dev interface]

Configuration d'une station

- **La commande netstat**

La commande **netstat**, permet de tester la configuration du réseau, visualiser l'état des connexions, établir des statistiques, notamment pour surveiller les serveurs.

```
farid@Serveur:/etc$ netstat –a
Proto RefCnt Flags       Type       State         I-Node   Path
unix  2      [ ACC ]     STREAM     LISTENING     11774    /tmp/orbit-farid/linc-64f-0-7fdc879498989
unix  2      [ ACC ]     STREAM     LISTENING     11792    /tmp/orbit-farid/linc-64c-0-411483fe46585
udp        0      0 Serveur.local:domain    *:*
udp        0      0 localhost:domain        *:*
tcp        0      0 Serveur.local:domain    *:*                 LISTEN
tcp        0      0 localhost:domain        *:*                 LISTEN
```

La commande **netsat –nr** ou **–r** permet aussi d'affiche l'état des tables de routage à l'instar de la commande **route -n**

Configuration d'une station

- **La commande traceroute**

La commande **traceroute** permet d'afficher le chemin parcouru par un paquet pour arriver à destination. Cette commande est importante, car elle permet d'équilibrer la charge d'une réseau en optimisant les routes.

```
farid@Serveur:/etc$ traceroute 192.168.1.1
traceroute to 192.168.1.1 (192.168.1.1), 30 hops max, 60 byte packets
 1  192.168.226.2 (192.168.226.2)  0.437 ms  0.089 ms  0.081 ms
 2  sagembox.home (192.168.1.1)  5.396 ms  6.799 ms  16.286 ms
```

Un administrateur, responsable d'un réseau d'entreprise sur lequel il y'a de nombreux routeurs, peut, avec cet outil, diagnostiquer les routes et temps de routage. Il peut ainsi optimiser les trajets et temps de réponse.

Configuration d'une station

- **Gérer la configuration dans le fichier interfaces**

  ➢La configuration d'une interface avec ifconfig n'est pas enregistrée sur le disque.

  ➢Pour enregistrer la configuration de manière permanente, il faut créer cette configuration dans le fichier de configuration /etc/network/interfaces..

Configuration d'une station

- **Gérer la configuration dans le fichier interfaces**

  ➢La configuration d'une interface avec ifconfig n'est pas enregistrée sur le disque.

  ➢Pour enregistrer la configuration de manière permanente, il faut créer cette configuration dans le fichier de configuration /etc/network/interfaces..

Configuration d'une station

- **Gérer la configuration dans le fichier interfaces**

  ➢Pour initialiser le réseau après configuration, il faut faire :

```
# /etc/init.d/networking start
```

  ➢Pour réinitialiser le réseau après un changement dans les fichiers de configuration, il faut faire :

```
# /etc/init.d/networking restart
```

Configuration d'une station

- **Gérer la configuration dans le fichier interfaces**

➢La configuration des interfaces utilisée lors de l'initialisation du réseau est contenue dans le fichier /etc/network/interfaces:

❖**Pour ethernet static**

 # cat /etc/network/interfaces

```
# configuration de l'interface lo (obligatoire)
auto lo
iface lo inet loopback
# configuration de l'interface eth0
auto eth0
iface eth0 inet static
address 192.168.0.2
netmask 255.255.255.0
```

Configuration d'une station

- **Gérer la configuration dans le fichier interfaces**

- Certains réseaux locaux sont configurés en *DHCP* (*pour Dynamic Host Configuration Protocol*).

- Dans ce cas, le fichier interfaces est nettement simplifié et ne dépend que de l'interface :

```
# cat /etc/network/interfaces
auto lo eth0
iface lo inet loopback
iface eth0 inet dhcp
```

Configuration d'une station

- **Configuration WIFI**
  - Le *WIFI* (protocole 802.11) est une technologie de réseaux locaux sans fil.
  - Les protocoles de cryptage de communications utilisés en WIFI sont : WEP et WPA
  - l'ESSID est le nom du réseau sans fil.

Configuration d'une station

- **Configuration WIFI**

```
# cat /etc/network/interfaces
auto lo
iface lo inet loopback
# l'interface eth1 correspond ici à la carte wifi
auto eth1
iface eth1 inet dhcp
wireless-essid mon_essid
wireless-mode managed
wireless-key AF32852BE7A39B522BG60C4353
```

Le *ESSID* et la clef *WEP* doivent correspondre et être correctement configurés sur le serveur sans fil.

Configuration d'une station

- **Configuration WIFI**

  Pour obtenir les paramètres spécifiques au réseau *WIFI*, on utilise iwconfig:

```
# iwconfig
lo        no wireless extensions.
eth0      no wireless extensions.
eth1      IEEE 802.11g ESSID:"my_essid"
          Mode:Managed Frequency:2.437 GHz Access Point: 00:F0:C3:A4:C1:41
          Bit Rate:54 Mb/s Tx-Power=20 dBm Sensitivity=8/0
          Retry limit:7 RTS thr:off Fragment thr:off
          Power Management:off
          Link Quality=0/100 Signal level=-23 dBm Noise level=-86 dBm
          Rx invalid nwid:0 Rx invalid crypt:0 Rx invalid frag:0
          Tx excessive retries:5 Invalid misc:2 Missed beacon:98
sit0      no wireless extensions.
```

Serveur *DHCP*

- **Introduction**
  - Un serveur DHCP (*Dynamic Host Configuration Protocol*) a pour rôle de distribuer des adresses IP à des clients pour une durée déterminée.
  - Le serveur DHCP alloue à un client, un bail d'accès au réseau, pour une durée déterminée (durée du bail). Le serveur passe en paramètres au client toutes les informations dont il a besoin (masque de sous réseau, serveur de noms, passerelle par défaut, nom du réseau).
  - Tous les nœuds critiques du réseau (serveur de nom primaire et secondaire, passerelle par défaut) ont une adresse IP statique ; en effet, si celle-ci variait, ce processus ne serait plus réalisable.

Serveur *DHCP*

- **Avantages de DHCP dans l'administration d'un réseau:**
  - Le protocole DHCP offre une configuration de réseau TCP/IP fiable et simple, empêche les conflits d'adresses et permet de **contrôler** l'utilisation des adresses IP de façon **centralisée**.
  - **économie d'adresse** : ce protocole est presque toujours utilisé par les fournisseurs d'accès Internet qui disposent d'un nombre d'adresses limité. Ainsi grâce à DHCP, seules les machines connectées en ligne ont une adresse IP.
  - Les postes itinérants sont plus faciles à gérer
  - Le changement de plan d'adressage se trouve facilité par le dynamisme d'attribution.

Serveur *DHCP*

- **Avantages de DHCP dans l'administration d'un réseau:**
  - Avec DHCP, il suffit d'attribuer une adresse au serveur. Lorsqu'un ordinateur client DHCP demande l'accès au réseau en TCP-IP son adresse est allouée dynamiquement à l'intérieur d'une plage d'adresses définie sur le serveur.
  - L'administrateur de réseau contrôle le mode d'attribution des adresses IP en spécifiant une **durée de bail** qui indique combien de temps l'hôte peut utiliser une configuration IP attribuée, avant de devoir solliciter le renouvellement du bail auprès du serveur DHCP.

Serveur *DHCP*

- **Avantages de DHCP dans l'administration d'un réseau:**
  - **L'adresse IP est libérée automatiquement, à l'expiration du bail**, pour un ordinateur client DHCP retiré d'un sous-réseau, et une nouvelle adresse est automatiquement définie pour ce dernier, lorsque cet ordinateur est reconnecté à un autre sous-réseau.
- **L'inconvénient:**
  - Le client utilise des trames de **broadcast** pour rechercher un serveur DHCP sur le réseau, cela charge le réseau. Si vous avez une entreprise avec plusieurs centaines de personnes qui ouvrent leur session le matin à 8 h ou l'après midi à 14 h, il peut s'en suivre de graves goulets d'étranglement sur le réseau. L'administrateur devra donc réfléchir sérieusement à l'organisation de son réseau.

Serveur *DHCP*

- **Fonctionnement de DHCP (Attribution d'une adresse DHCP)**
- Un client DHCP est un ordinateur qui demande une adresse IP à un serveur DHCP.
  - Lorsqu'un client DHCP initialise un accès à un réseau TCP/IP, le processus d'obtention du bail IP se déroule en 4 phases :

    1 - Le client émet un message de demande de bail IP (**DHCPDISCOVER**).

    2 - Les serveurs DHCP répondent en proposant une adresse IP avec une durée de bail et l'adresse IP du serveur DHCP (**DHCOFFER**)

    3 - Le client sélectionne la première adresse IP (s'il y a plusieurs serveurs DHCP) reçue et envoie une demande d'utilisation de cette adresse au serveur DHCP (**DHCPREQUEST**).

    4 - Le serveur DHCP accuse réception de la demande et accorde l'adresse en bail (**DHCPACK**).

Serveur *DHCP*

- **Fonctionnement de DHCP (Renouvellement de bail IP)**
  - Lorsqu'un client redémarre, il tente d'obtenir un bail pour la même adresse avec le serveur DHCP d'origine, en émettant un **DHCPREQUEST**. Si la tentative se solde par un échec, le client continue à utiliser la même adresse IP s'il lui reste du temps sur son bail.
  - Lorsque le bail expire ou qu'un message **DHCPNACK** est reçu le client doit cesser d'utiliser l'adresse IP et demander un nouveau bail (retour au processus de souscription). Lorsque le bail expire et que le client n'obtient pas d'autre adresse la communication TCP/IP s'interrompt.

Remarque : Si la demande n'aboutit pas et que le bail n'est pas expiré, le client continue à utiliser ses paramètres IP.

Serveur *DHCP*

- **Configuration d'un serveur *DHCP***
- Définir une plage d'adresses qui peuvent être louées à des hôtes qui en font la demande. En général on donne:
  - Une adresse de début (la première qui sera attribuée)
  - Une adresse de fin (la dernière)
  - Une ou plusieurs plages d'adresses à exclure de la location (ceci permet de faire cohabiter un modèle de configuration IP dynamique avec un modèle statique)
  - Un masque de sous-réseau
- Tous ces éléments sont attribués pour une durée de bail à fixer. Si, au bout de cette durée, l'hôte ne sollicite pas à nouveau une adresse au serveur, cette adresse est jugée disponible pour un autre hôte.

Serveur *DHCP*

- **Configuration d'un serveur *DHCP***
- Pour redémarrer les serveur *DHCP* après un changement de configuration, on dispose de la commande : **# /etc/init.d/dhcp3-server restart**

Le fichier de configuration du serveur dhcp3-server est : /etc/dhcp3/dhcpd.conf

- **option domain-name** "monserveur.com" : Nom de domaine correspondant au réseau local.
- **option domain-name-servers:** adresse IP serveurs DNS.
- **subnet:** Donne une idée au serveur *DHCP* de la topologie du réseau.
- **default-lease-time** : temps de renouvellement de bail.
- **option routers**: adresse routeur/passerelle
- **option ntp-servers**: serveur de temps.

Serveur *DHCP*

Configuration d'un serveur *DHCP*

Exemple :Editer le fichier /etc/dhcp/dhcpd.conf

…….

```
subnet 192.168.0.0 netmask 255.255.255.0 {
range 192.168.0.2 192.168.0.20;
option routers 192.168.0.1;
default-lease-time 600;
max-lease-time 7200;
}
```

Pour réserver une adresse IP fixe particulière à un certain client identifié par son adresse MAC .

```
host guest {
hardware ethernet 67:42:AB:E3:74:00;
fixed-address 192.168.0.3;
}
```

….

Serveur *DHCP*

- **Mise en œuvre d'un client DHCP**
- Les clients DHCP doivent être configurés seulement après la configuration du serveur. Etant donné qu'un ordinateur ne peut fonctionner simultanément comme client et serveur DHCP, l'ordinateur fonctionnant comme serveur DHCP doit être configuré avec une adresse IP fixe.
- Modifier la configuration de l'interface dans le fichier /etc/network/interfaces

```
# cat /etc/network/interfaces
auto lo eth0
iface lo inet loopback
iface eth0 inet dhcp
```

  - Le client DHCP lance un processus **dhclient** qui cherche à contacter un serveur DHCP (sudo dhclient eth0 utiliser l'option –r pour forcer).

Serveur *DHCP*

- **Rôle de l'agent de relais DHCP**
- dans un inter-réseau, vous devrez théoriquement installer un serveur DHCP par sous-réseau.
- Le routeur peut faire office d'agent de relais dhcp, et ainsi relayer les diffusions de demande d'adresse IP des clients DHCP dans chaque sous-réseau.

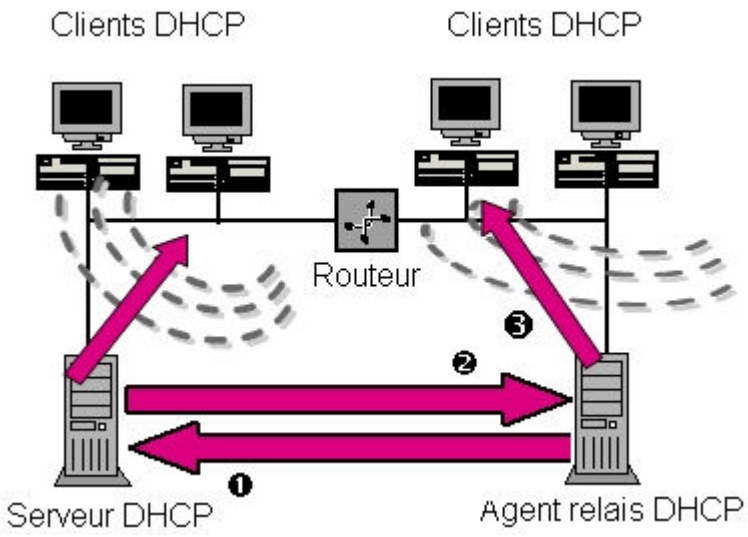


Fig: Agent de relais DHCP dans un réseau routé

- Une machine serveur peut être configurée comme agent de relais DHCP il suffira de lui spécifier l'adresse du serveur DHCP.
- Les demandes des clients DHCP seront relayées vers le serveur DHCP par l'agent de relais DHCP qui transmettra les offres aux clients.

Serveur DNS

- **Introduction** (Historique)
  - Chaque ordinateur du réseau contenait un fichier **/etc/hosts.**

    Problème: À chaque fois que l'on rajoutait une machine sur le réseau, il fallait mettre à jour ce fichier.
  - La solution qui s'est imposée fut la création d'une base de données distribuée, et ainsi est né le principe de serveur DNS .
  - Un serveur DNS permet de faire la correspondance entre un nom canonique **FQDN** (ex: www.google.fr) et son adresse IP.
  - **BIND** est le serveur DNS le plus utilisé et le plus populaire, environ 70% d'Internet fonctionne avec ce logiciel.

Serveur DNS

- **Introduction** (Historique)
  - Le DNS est basé sur un modèle en arborescence similaire à celui des systèmes de fichiers et de répertoires, avec une gestion décentralisée des données .
  - Il permet de fournir des informations supplémentaires telles que celles concernant le temps de validité des informations, les relais de messagerie, les alias de machines, etc..., et assure une mise en correspondance dynamique entre les noms et les adresses IP.
  - Un système de réplication assure une fiabilité raisonnable, tandis qu'un système de caches permet d'augmenter la performance du système.

Serveur DNS

- **Introduction** (Qu'est-ce qu'un nom de domaine ?)
  - Le **nom de domaine** est une partie intégrante de l'adresse de toute ressource Internet. Il en constitue même l'élément fondamental.
  - Dans le système hiérarchique du DNS, le nom de domaine constitue une entité qui permet la localisation du domaine, à l'instar de celle d'un répertoire dans un système de fichiers.
  - Le nom de domaine est constitué d'éléments séparés par un "." (analogie avec le "/" ou "\" dans un système de fichiers pour localiser un répertoire).

Serveur DNS

- **Introduction** (Qu'est-ce qu'un nom de domaine ?)
  - Dans un réseau local, les machines peuvent-être identifiées par leurs seuls noms. Par contre, à l'échelle d'Internet, ces noms doivent être concaténés avec le nom du domaine dans lequel elles sont déclarés.

    Exemple:

    Nom du domaine : **afnic.fr**
    nom local de la machine : **www**
    nom de la machine dans le DNS : **www.afnic.fr**

    nom local de la machine : **ftp**
    nom de la machine dans le DNS : **ftp.afnic.fr**

Serveur DNS

- **Introduction** (Qu'est-ce qu'un nom de domaine ?)
  - Les informations sur la correspondance entre nom et adresse IP sont stockées dans la base de données des différents domaines.
  - Pour l'exemple précédent, la base de données du domaine afnic.fr va contenir des informations du type :
    - nom de machine :**ftp.afnic.fr** => adresse IP :**192.134.4.13**
    - nom de machine :**relay1.afnic.fr** => adresse IP :**192.134.4.17**
    - nom de machine :**www.afnic.fr** => adresse IP :**192.134.4.11**
    - adresse IP **:192.134.4.11** => nom de machine **:www.afnic.fr**
  - la dernière entrée correspond à une résolution inverse; elle spécifie notamment que l'adresse IP 192.134.4.11 est celle d'une machine à laquelle est attribué le nom www.afnic.fr.

Serveur DNS

- **Principe de fonctionnement du DNS (Arbre de nommage)**
  - Le **DNS** est organisé sous forme d'un arbre renversé avec comme éléments :
    - la **racine** (root) qui constitue le sommet de l'arbre ; elle est représenté par un point "." ;
    - des **nœuds** qui représentent des domaines et sont identifiés chacun par un label (exemple : fr, nl, sn, com, etc...) .
  - Au sommet de l'arbre on trouve des serveurs root qui aiguille vers les top level domain (com, net org,fr,etc...)
  - Il existe une branche spéciale **ARPA** avec un sous domaine in-addr qui sert à gérer le reverse DNS.

Serveur DNS

- **Principe de fonctionnement du DNS (Arbre de nommage)**

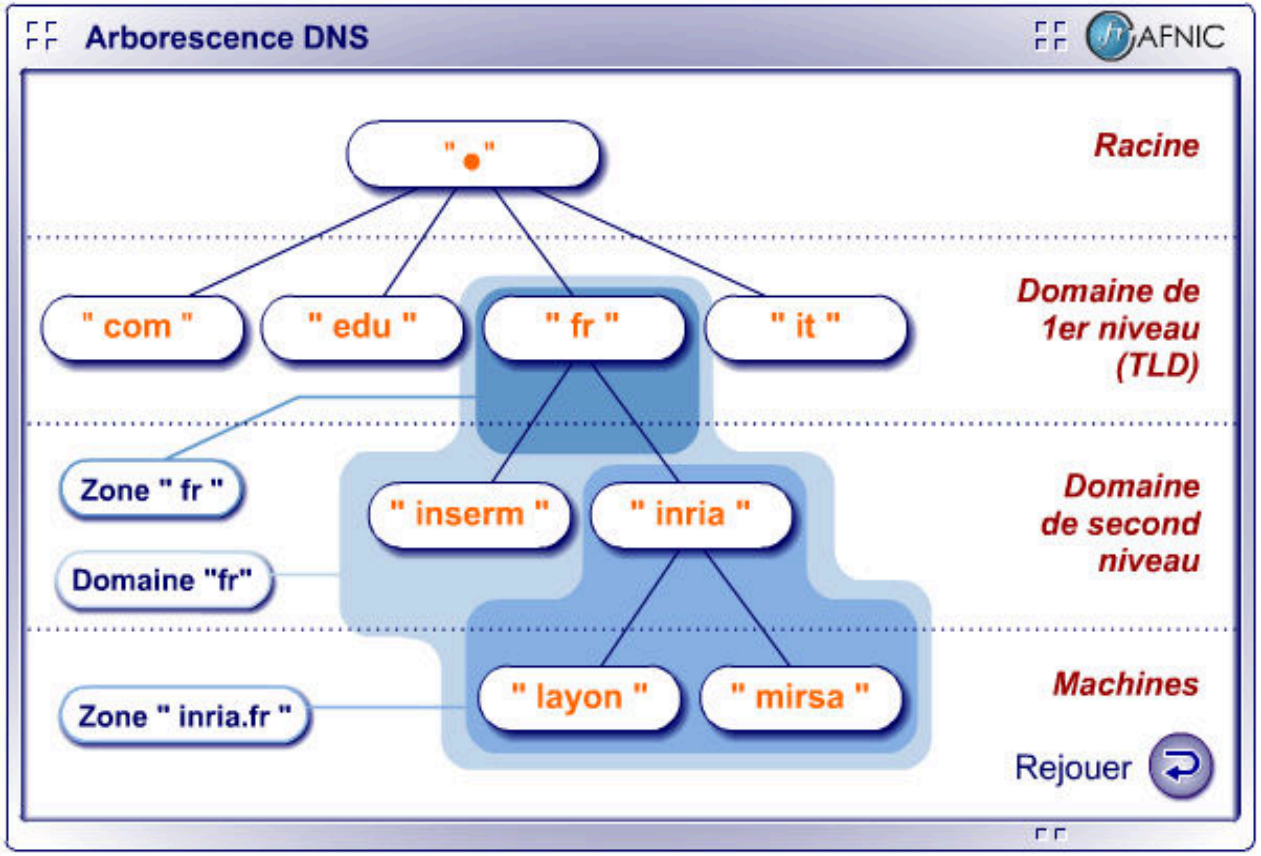


Schéma arbre à l'envers (source www.afnic.fr)

Serveur DNS

- **Principe de fonctionnement du DNS (Domaine et zone)**
  - Un domaine représente l'ensemble d'une sous-arborescence à partir d'un nœud donné.
  - Chaque nœud de l'arbre de nommage est un domaine. En dehors de la racine, chaque domaine peut-être considéré comme un sous-domaine pouvant lui-même contenir des sous-domaines.
  - Dans l'exemple d'arborescence précédent, le domaine "**fr**" englobe les sous-domaines "**inria**" et "**inserm**". Une zone peut correspondre à un domaine, mais dans le cas général, il englobe uniquement une partie du domaine, le reste étant délégué à d'autres serveurs de noms.

Serveur DNS

- **Principe de fonctionnement du DNS (**Résolution de nom et résolution inverse)
  - Prenons un cas pratique : résoudre le nom d'hôte **electron.nic.fr.**
    1. La machine cherchant à atteindre cet hôte contacte l'un des serveurs de noms par défaut (**3** au maximum).
    2. Si ce serveur de noms par défaut n'arrive pas à résoudre ce nom, il contacte les serveurs de noms à la racine. Le DNS local interroge alors le DNS de la zone **.fr**.
    3. Si ce serveur de noms n'est capable de résoudre **electron.nic.fr.**, il renvoie la liste des serveurs de noms de la zone **nic.fr**.le serveur de noms de la zone **nic.fr**, connaît alors l'adresse IP de **electron.nic.fr** (**192.134.4.9**)

Serveur DNS

- **Principe de fonctionnement du DNS (**Résolution de nom et résolution inverse)
  - La résolution inverse consiste, elle, à retrouver le nom d'une machine à partir de son adresse IP.
  - Comme pour la résolution de nom, la résolution inverse s'appuie sur un système en arborescence qui part du domaine particulier **in-addr.arpa**
    1. Chaque octet de l'adresse IP correspond à un niveau ; pour chaque adresse, on crée un nom de domaine, sous-domaine de **in-addr.arpa**, qu'on obtient en écrivant l'adresse IP ‘’à l'envers" (en commençant par le dernier octet). Ainsi, pour l'adresse IP **192.134.4.9**, on obtient le nom de domaine **9.4.134.192.in-addr.arpa**.
    2. Puis la procédure est le même que celle de la résolution classique.

Serveur DNS

- **Architecture (Modèle client/serveur)**
  - Pour interroger un serveur de noms, on n'a pas besoin d'avoir de DNS local. Pour toute requête, on passe par une bibliothèque appelée "resolver"
  - Le **resolver** est un ensemble de fonctions écrites en C (comme *gethostbyname*, par exemple).
  - Sur un serveur Unix, Le fichier de configuration du resolver s'appelle /etc/resolv.conf.

```
$ cat /etc/resolv.conf
   search mondomaine.fr
   nameserver 192.168.30.1
```

  - L'enregistrement ''search" spécifie le domaine par défaut (à ajouter aux noms de machines ci ceux-ci ne sont pas donnés en **FQDN**) ;

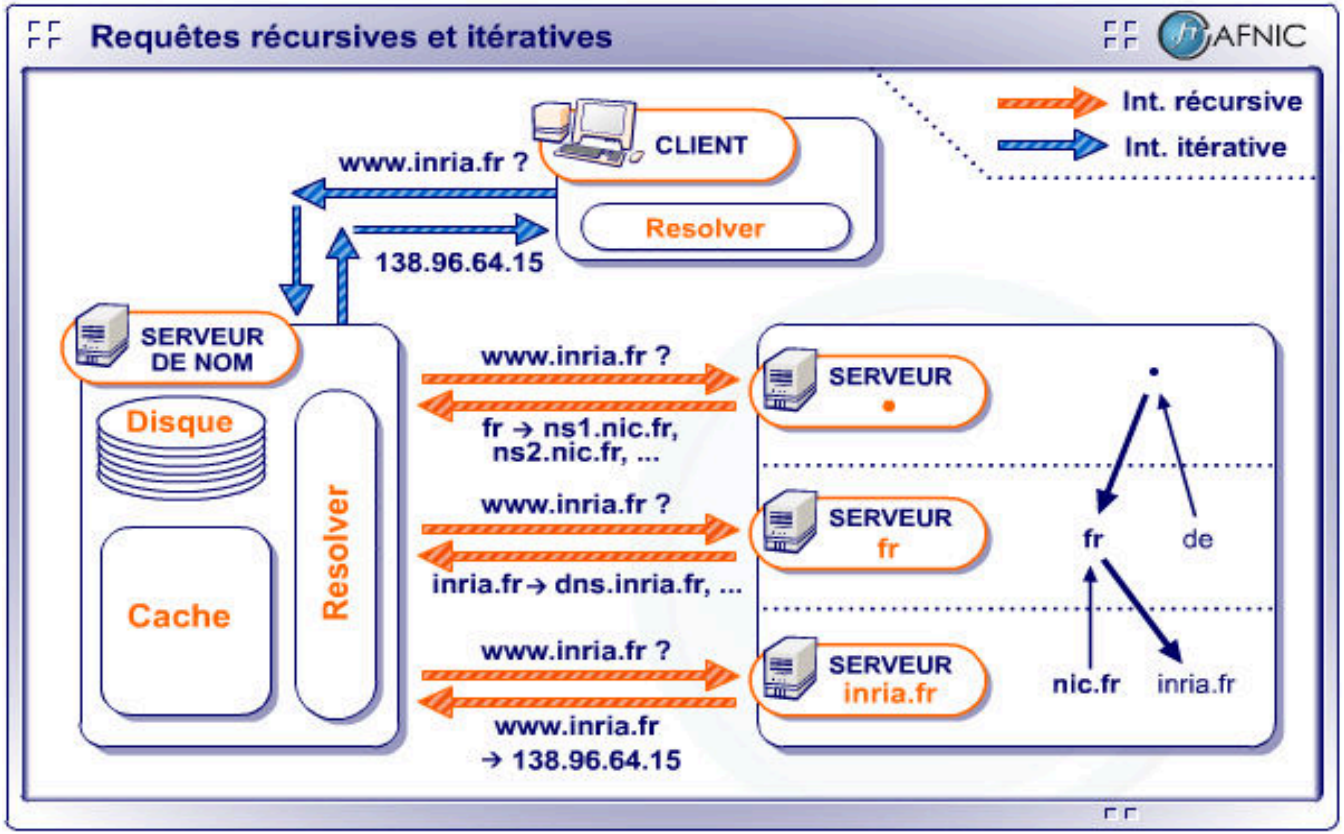
Serveur DNS

- **Architecture (Client)**
  - Le resolver permet de lancer des requêtes DNS pour la résolution de nom. Il existe 2 modes d'interrogation pour un résolver :
    - Le **mode récursif** : le **client** (resolver) envoie une requête au **serveur DNS** ; ce dernier renvoie une réponse complète au client qui est soit la correspondance recherchée soit un message d'erreur.
    - Le **mode non récursif** ou **itératif** : le client envoie une requête au serveur DNS ; ce dernier renvoie soit la réponse complète (s'il est autoritaire pour la zone concernée) soit une réponse partielle (adresse d'un autre serveur de noms qui va permettre au client d'avancer dans le processus de résolution).

En général, le **mode récursif** est utilisé par les **applications clientes** et le **mode itératif** par les **revolvers des serveurs de noms**.

Serveur DNS

- **Architecture (Client)**



Requêtes récursives et itératives (source www.afnic.fr)

Serveur DNS

- **Architecture (Serveur cache)**
  - Les serveurs de noms qui se chargent de traiter des requêtes récursives sont appelés des **serveurs de noms récursifs**. Afin d'améliorer leurs performances et éviter la surcharge du réseau et des autres serveurs DNS, les serveurs récursifs utilisent un cache pour stocker les réponses des requêtes.
  - Pour les requêtes ultérieures des clients, un serveur cache consulte d'abord son cache pour voir si la requête n'avait pas déjà été traitée (par exemple pour un autre client). Si c'est le cas, les données correspondantes du cache sont fournies en réponse.
  - Une configuration minimale d'un serveur cache contient la liste des **serveurs de la racine** (serveurs root) ainsi que l'enregistrement pour le **reverse du loopback** (0.0.127.in-addr.arpa).

Serveur DNS

- **Architecture (Serveur forward)**
  - Si la requête ne peut pas être satisfaite avec les données du cache, un serveur DNS peut faire suivre la requête à un autre serveur de noms. Dans le cas ou il fait suivre systématiquement les requêtes à d'autres serveurs, il est qualifié de serveur "**faire-suivre" esclave (ou forwarder esclave)**.
  - Le serveur auquel on fait suivre les requêtes est aussi appelé **forwarder**. Si l'information requise n'est pas dans son cache, il contacte soit d'autres **forwarders**, soit d'autres **serveurs autoritaires** (voir le paragraphe suivant).

Serveur DNS

- **Architecture (Serveur autoritaire)**
  - Dans l'arbre de nommage, une zone est associée à chaque nœud qui correspond, lui, à un domaine. Le serveur de nom dans lequel est stocké la base de données de la zone est dit "**faisant autorité sur la zone**". Il est aussi appelé **serveur autoritaire**.
  - Compte tenu de l'importance de la base de données de zone, il y a en général plusieurs serveurs autoritaires pour chaque zone (voir aussi le cas des serveurs root). Cependant, les enregistrements ne sont modifiés pour les mises à jour que sur un seul serveur autoritaire de la zone. Ce serveur est appelé **serveur primaire** ; on dit qu'il a l'origine de l'autorité sur la zone (**SOA** : **S**tart **O**f **A**utorithy).
  - Les autres serveurs autoritaires de la zone sont appelés **serveurs secondaires** et disposent chacune d'une copie de la base de données du serveur primaire. Ces copies sont mises à jour régulièrement suivant un mécanisme appelé **transfert de zone**.

Serveur DNS

- **Architecture (Rafraichissement des données)**
  - Les modifications de la base de données d'une zone ne sont effectuées qu'au niveau du serveur primaire; la mise à jour des copies des serveurs secondaires s'effectue par un transfert de zone classique, soit sur la base d'une notification du serveur primaire.
  - **Transfert de zone classique :** les serveurs secondaires de la zone interrogent à intervalles réguliers le serveur primaire afin de savoir s'il y a eu des modifications dans la base. La fréquence de cette interrogation est spécifiée par la valeur du paramètre **refresh** défini dans l'enregistrement **SOA** de la base de données. Ce mode de transfert peut s'effectuer aussi bien entre un serveur primaire et un serveur secondaire qu'entre deux serveurs secondaires d'une même zone.

Serveur DNS

- **Architecture (Rafraichissement des données)**
  - Si la phase d'interrogation aboutit à un échec, le serveur secondaire recommence la procédure toutes les **x** secondes, **x** étant un temps défini dans le paramètre **retry** de l'enregistrement **SOA**.
  - Si la procédure n'aboutit pas après un temps défini dans le paramètre **expire** de l'enregistrement SOA, le secondaire abandonne la procédure de transfert de zone en cours et ne répond plus aux requêtes en tant que serveur autoritaire pour cette zone.
  - **DNS Change Notification :** cette technique de transfert de zone est basée sur la notification par le serveur primaire à tous les serveurs secondaires de modifications effectuées sur le fichier de zone.

Serveur DNS

- **Introduction à la configuration (Fichiers de zones)**
  - Le fonctionnement du DNS requiert l'existence d'un certain nombre de fichiers au niveau de chaque serveur autoritaire:
    - Un fichier de configuration du serveur.
    - Un fichier des serveurs racines.
    - Un fichier par zone pour toutes les zones pour lesquelles le serveur est autoritaire.
  - La configuration d'un serveur DNS fait intervenir plusieurs paramètres dont la définition est donnée ci-après:
    - RR (Ressource Record) L'enregistrement de ressource peut être considéré comme l'unité de donnée de base du DNS. Il donne certaines caractéristiques d'un nom de domaine telles qu'une adresse IP d'un serveur ou d'une machine du domaine, un alias pour une machine, etc...

Serveur DNS

Introduction à la configuration (Fichiers de zones)

Un RR est composé de cinq éléments : un **FQDN**, un **TTL**, une **classe d'enregistrement**, un **type d'enregistrement** et une **valeur** ;

- **TTL (Time To Live)** : La durée de vie est le temps pendant lequel la valeur d'un RR dans un cache est considérée comme valide ;
- **Record Class** : La classe d'enregistrement définit le schéma de base sur lequel est fondé le système d'enregistrements. Il en existe trois : **Internet (IN)**, **Hesiod** (utilisé un certain temps par le M.I.T.) et **Chaos** (obsolète). La quasi-totalité des serveurs DNS (plus de 99%) fonctionnent avec la classe Internet ;
- **Record Type** : Donne le type d'un enregistrement (adresse IP, alias, nom de domaine, etc...) ;
- **SOA (Start Of Authority)** : L'origine de l'autorité spécifie la zone pour laquelle le serveur est autoritaire ;

Serveur DNS

Introduction à la configuration (Fichiers de zones)

- **NS (Name Server)** : NS est un type d'enregistrement qui spécifie le serveur autoritaire pour la zone considérée ;
- **A (Address)** : Permet la mise en correspondance entre un FQDN et une adresse IP;
- **CNAME (Canonical Name)** : Le type d'enregistrement nom canonique donne un alias pour une machine donnée ;
- **MX (Mail eXchanger)** : Ce type d'enregistrement spécifie le relais de messagerie pour la zone considéré ;
- **PTR (Pointer)** : Le type d'enregistrement "Pointeur" est utilisé pour la résolution inverse pour la mise en correspondance entre une adresse IP et un nom de domaine.

Serveur DNS

- **Les fichiers de configuration du client :**
- Il faut d'abord modifier deux fichiers pour indiquer quels serveurs de noms utiliser, et quels services de conversion de noms sont disponibles.
  - Le fichier **/etc/resolv.conf** détermine la façon dont le DNS doit chercher les informations :

```
#.....
nameserver 127.0.0.1
```

  - Le fichier **/etc/host.conf** indique quels services de conversion de noms sont disponibles, et dans quel ordre il faut les appliquer :

```
#.....
order hosts, bind
```

Serveur DNS

- **Configuration d'un serveur DNS (BIND)**
- Le fichier principal de configuration de BIND est /etc/bind/named.conf
- Debian a choisit de découper ce fichier en 3 fichiers afin de faciliter les mises à jour.
  - /etc/bind/named.conf
  - /etc/bind/named.conf.options
  - /etc/bind/named.conf.local

  A noter que dans ces fichiers, les lignes en commentaire commencent par // et non le **#** habituel.

Serveur DNS

- **Configuration d'un serveur DNS (BIND)**

a) Fichier de zone directe

- On va configurer le DNS pour la zone directe **/etc/bind/db.mondomaine.fr**
- On rajoute tout d'abord la zone dans le fichier **named.conf.local** :

```
// La zone directe DNS

  zone "db.mondomaine.fr" {

  type master;

   file "/etc/bind/db.mondomaine";

   };
```

Serveur DNS

- **Configuration d'un serveur DNS (BIND)**

a) Fichier de zone directe

- On crée ensuite le fichier de zone **db.mondomaine.fr**

```
# cat /etc/bind/db.mondomaine
;
; BIND data file for mondomaine.fr
;
$TTL    604800
@       IN      SOA     dns.mondomaine.fr.
root.mondomaine.fr. (
                        1         ; Serial
                   604800         ; Refresh
                    86400         ; Retry
                  2419200         ; Expire
                   604800 )       ; Negative Cache TTL
;
@           IN      NS              dns.mondomaine.fr.
pc210       IN      A               192.168.30.210
srv1        IN      A               192.168.30.211
```

Serveur DNS

- **Configuration d'un serveur DNS (BIND)**

a) Fichier de zone directe

- Les lignes en commentaire commencent par ; et non le # habituel.
- Points importants de ce fichier :
  - Le caractère @ (arobas) remplace le nom de la zone.
  - Lorsque l'on définit un nom canonique, on a deux possibilités
    - Soit on donne le nom en entier (ex: pc210.mondomaine.fr.).
    - Soit on ne donne que le nom "court" (ex: alex).

Serveur DNS

- **Configuration d'un serveur DNS (BIND)**

b) Fichier de zone inverse

- Maintenant que l'on a configuré le DNS de la zone **mondomaine.fr**, on va créer la zone inverse qui va permettre d'associer un nom à une adresse IP.
- On rajoute tout d'abord la zone inverse dans le fichier **named.conf.local** :

```
// La zone reverse DNS
zone "30.168.192.in-addr.arpa" {
type master;
 file "/etc/bind/db.192.168.30";
 };
```

Serveur DNS

- **Configuration d'un serveur DNS (BIND)**

b) Fichier de zone inverse

- Le nom de la zone est composé de l'adresse réseau (à l'envers) associé à **in-addr.arpa.**
- On crée ensuite le fichier **/etc/bind/db.192.168.30** :

```
# cat /etc/bind/db.192.168.30
;
; BIND data file for 192.168.30
;
$TTL    604800
@       IN      SOA     dns.mondomaine.fr.
root.mondomaine.fr. (
                              1         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@       IN      NS      dns.mondomaine.fr.
210     IN      PTR     pc210.mondomaine.fr.
211     IN      PTR     srv1.mondomaine.fr.
```

Serveur DNS

- **Test de fonctionnement**
- Une fois que l'on a modifié ces fichiers, on relance le serveur DNS :

```
# /etc/init.d/bind restart
```

- On modifie ensuite le fichier **/etc/resolv.conf** pour lui indiquer d'utiliser le DNS que l'on vient de configurer :

```
# vi /etc/resolv.conf
search mondomaine.fr
nameserver 127.0.0.1
```

- Pour tester, on essaye de pinger un nom définit dans le DNS

```
# ping pc210.mondomaine.fr
```

Si tout se passe bien, le DNS doit effectuer la résolution.

  - On peut aussi utiliser les commandes **host** et **dig** pour vérifier

Serveur DNS

- **Test de fonctionnement (Commandes `host`)**
  - La commande host permet d'obtenir l'adresse IP d'un ordinateur :

```
$ host www.google.com
www.google.com is an alias for www.l.google.com.
www.l.google.com has address 209.85.135.147
www.l.google.com has address 209.85.135.99
www.l.google.com has address 209.85.135.103
www.l.google.com has address 209.85.135.104
```

Serveur DNS

- **Test de fonctionnement (Commandes host)**
  - La commande **host** permet également de consulter le DNS inverse, c'est à dire quel nom canonique est associé à une adresse IP donnée :

```
$ host 66.249.93.104
104.93.249.66.in-addr.arpa domain name pointer ug-in-f104.google.com.
```

Serveur DNS

- **Commandes `dig`**
  - La commande **dig** permet d'interroger un serveur DNS
  - Voici quelques exemples :

    En interrogeant le sommet de l'arbre , on obtient la liste des serveurs racines du DNS, appelles les root servers

```
$ dig . NS
... ;; ANSWER SECTION:
.       419748 IN NS a.root-servers.net.
.       419748 IN NS b.root-servers.net.
.       419748 IN NS c.root-servers.net.
.       419748 IN NS d.root-servers.net.
...
```

Serveur DNS

- **Test de fonctionnement (Commandes `dig`)**
  - En interrogeant la branche **com**, on obtient la liste des serveurs DNS gérant les noms de domaines en .com

```
$ dig com. NS

 ...

;; ANSWER SECTION:

com. 172800 IN NS i.gtld-servers.net.

com. 172800 IN NS j.gtld-servers.net.

com. 172800 IN NS k.gtld-servers.net.

com. 172800 IN NS l.gtld-servers.net.

...
```

Serveur DNS

- **Test de fonctionnement (Commandes dig)**
  - En interrogeant la branche **com**, on obtient la liste des serveurs DNS gérant les noms de domaines en .com

```
$ dig com. NS

 ...

;; ANSWER SECTION:

com. 172800 IN NS i.gtld-servers.net.

com. 172800 IN NS j.gtld-servers.net.

com. 172800 IN NS k.gtld-servers.net.

com. 172800 IN NS l.gtld-servers.net.

...
```

Serveur DNS

- **Test de fonctionnement (Commande nslookup)**
  - La commande nslookup permet elle aussi d'interroger la zone dns de votre domaine.
  - La structure générale de la commande nslookup est:

```
$ nslookup
$ set type=<type RR>
```

avec type RR=(Any, A, NS, PTR, MX, CNAME …)

  - La commande nslookup étant obsolète, il est préférable pour vérifier et diagnostiquer DNS via la commande dig

OFPPT

2ème partie:
Partage de fichiers

Partage de fichiers

Installation de *NFS*

Le protocole ***NFS*** (*Network File System*) est un modèle client-serveur qui permet de partager facilement des fichiers entre des machines *Unix*, et donc *Linux*.

❖ Installation NFS coté serveur

- **S**ur le serveur, il faut installer **portmap**, **nfs-common**, et **nfs-kernel-server**.
- Sur le client, il est faut installer **nfs-common** et **portmap**.
- En ce qui concerne les sécurités, sachez que NFS utilise le **wrapper tcp** (**tcpd**). Il est possible de configurer la sécurité via les fichiers /etc/hosts.allow et /etc/hosts.deny.
- Les protocoles à ouvrir sur le serveur sont **statd**, **nfsd**, **lockd**, **rquotad** et **mountd**. Sur le client, il faut permettre à **statd** d'accèder à localhost

Partage de fichiers

- **Installation de *NFS***
  - Les programmes sur lequel s'appuie le service NFS utilisent les **RPC** (Remote Procedure Call). Ils s'inscrivent donc auprès du service **portmap** qui met à jour sa table de service **rpc**. Voici ci-contre un extrait de ce que donne la commande **rpcinfo –p.**
  - les processus qui doivent être actifs sur le serveur NFS.

```
program vers proto   port
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
    100003    2   udp   2049  nfs
    100003    3   udp   2049  nfs
    100003    2   tcp   2049  nfs
    100003    3   tcp   2049  nfs
    100021    1   udp  33065  nlockmgr
    100021    3   udp  33065  nlockmgr
    100021    4   udp  33065  nlockmgr
    100021    1   tcp  38399  nlockmgr
    100021    3   tcp  38399  nlockmgr
    100021    4   tcp  38399  nlockmgr
    100005    1   udp    967  mountd
    100005    1   tcp    970  mountd
    100005    2   udp    967  mountd
    100005    2   tcp    970  mountd
    100005    3   udp    967  mountd
    100005    3   tcp    970  mountd
```

portmap gère le catalogue des programmes RPC,

mountd est chargé des opérations de montage/démontage d'arborescence,

nfsd exécute les primitives d'accès aux fichiers - requêtes émanant des clients.

Partage de fichiers

- **Lancement du serveur NFS**

Vous pouvez lancer les démons *NFS* manuellement avec :

```
\# /etc/init.d/nfs-kernel-server start
```

Pour redémarer le serveur *NFS* proprement :

```
\# /etc/init.d/nfs-kernel-server restart
```

On peut vérifier que les démons sont lancés :

```
/etc/init.d/nfs-kernel-server status
```

Nous pouvons aussi relancer le service par :

```
service nfs-kernel-server restart|reload
```

Partage de fichiers

- **Configuration du serveur NFS**

Le fichier de configuration du serveur *NFS* est /etc/exports.

Exemple :

```
/home ollinux(rw) station1(ro)\\
/projet station1(rw) (ro)\\
/brouillon\\
```

ollinux pourra monter le dossier /home en lecture/écriture (rw).

station1 pourra monter le dossier /home en lecture (ro).

Enfin tout le monde pourra accéder en lecture/écriture au répertoire brouillon

(l'option rw est celle par défaut).

Nous relançons le service *NFS* après config par la commande suivante:

```
\textit{/etc/init.d/nfs-kernel-server reload}
```

Partage de fichiers

Configuration du client NFS

Il n'y a pas de fichier particulier. Pensez juste à lancer le **portmap**, sinon le montage restera en attente (`$ /etc/init.d/portmap start`).

Pour monter un système de fichier distant, utiliser la commande **mount** avec l'option nfs

```
$ mount -t nfs machine distante: répertoire\_partagé répertoire\_local -o options
```

Par exemple :

```
$ mount -t nfs 192.168.105.2:/armor/plages /mnt/cotes -o ro\\
```

Partage de fichiers

- **Fichier /etc/fstab**

Afin de se connecter aux répertoires partagés au démarrage de la station, Le plus simple est de renseigner le fichier **/etc/fstab**.

La syntaxe est la suivante :

```
ordinateur-distant:répertoire-distant répertoire-local nfs
options 0 0
```

Pour reprendre l'exemple précédent, cela donnerait :

```
monhost:/armor/plages /mnt/cotes nfs auto,rw,user,soft 0 0
```

Partage de fichiers

Installation de *Samba*

- **Samba** est un service permettant de partager des dossiers et des imprimantes entre les ordinateurs d'un réseau local fonctionnant avec des systèmes d'exploitation Microsoft Windows et Apple Mac OS, ainsi que des systèmes GNU/Linux, *BSD et Solaris dans lesquels une implémentation de Samba est installée.
- Le fichier de configuration du serveur **Samba** est smb.conf qui se trouve généralement dans le répertoire /**etc**/**samba** (ou /**etc** selon la distribution).
- Nous étudions ici le simple partage de répertoires avec authentification par login et mot de passe.
- Le paquet à installer s'appel *samba.*

Partage de fichiers

Installation de *Samba*

- Le paquet installe principalement samba et samba-common :
  - le programme **nmbd** qui assure la résolution de nom **NetBIOS** et **smbd** qui assure le partage de ressource **SMB/CIFS** dans /**usr/sbin**,
  - le script d'initialisation dans /**etc/init.d**,
  - un fichier de configuration /**etc/samba/smb.conf**,
  - une documentation complète dans /**usr/share/doc**,
  - le service de journalisation (log) dans /**var/log/samba**,
  - des outils comme **smbpasswd** pour la création des comptes samba et **nmblookup** pour vérifier le fonctionnement de la résolution de noms NetBIOS.

Partage de fichiers

Le fichier de configuration sous Linux

- Le fichier de configuration va permettre de :
  - définir NomDuServeur comme serveur Samba,
  - mettre en place l'authentification des utilisateurs,
  - partager des disques et une imprimante pour un client Windows,
  - partager le dossier personnel d'un utilisateur sous Linux comme étant son répertoire personnel sous Windows.
- Le fichier de configuration comprend essentiellement deux parties :
  - une partie “ **générale** ” qui définit le comportement général du serveur et la stratégie adoptée pour les services communs (CPD, mode d'authentification, service WINS)...
  - une partie **share**, qui définit les ressources partagées et les permissions d'accès.

Partage de fichiers

Les étapes de la configuration du serveur

- Nous allons réaliser les opérations suivantes :
  - Vérifier et valider le fichier de configuration à l'aide de la commande `testparm|more` .
  - Déclarer les ressources partagées,
  - Créer des comptes utilisateurs pour SAMBA.
- Puis, Il n'y aura plus qu'à tester la configuration à partir d'un client.

Attention, un compte système n'est pas un compte SAMBA. Faites bien la distinction entre les deux.

Partage de fichiers

Fichier de configuration d'un serveur SAMBA :

- le fichier de configuration est smb.conf, la configuration générale est la suivante :

• **workgroup =** nom du groupe de travail Windows (qui est en général Workgroup par défaut sous Windows).

netbios name = Nom du PC sur le réseau.

• **server string =** Description de votre pc que vous voulez donner

• **security =** user pour que seuls les titulaires d'un compte *Unix* sur le serveur puissent utiliser *samba,* ***SHARE****: tout le monde*.

• **encrypt passwords =** true (pour activer le cryptage de password. Attention au s de passwords)

• **log file =** /var/log/samba/log.%m (chemin du fichier log)

• **max log size =** 50 (taille du fichier log)

• **socket options =** TCP NODELAY (permet d'avoir de meilleur performance réseaux)

• **dns proxy =** No (permet d'activer la fonction de résolution de noms NetBios)

Partage de fichiers

- **Fichier de configuration d'un serveur SAMBA :**

- Quelques options peuvent de plus être ajoutées :

printcap name = /etc/printcap (chemin du fichier qui contient la liste des imprimantes)
load printers = yes (va avec printcap et permet de charger toutes les imprimantes installé sur le system, ça évite de les configurer une à une dans le smb.conf)

- D'autres options sont possibles pour par exemple configurer samba pour qu'il utilise un serveur windows Nt comme domaine primaire, ou pour qu'il résolve les noms d'hôtes, etc...

Partage de fichiers

- **Fichier de configuration d'un serveur SAMBA :**
  - Configuration du partage de répertoires (partie share):

```
[homes]
# permet de partager le répertoire personnel de chaque utilisateur
comment = Home Directories
browseable = no
writable = yes
create mask = 0755
directory mask = 0755

[partage]
comment = Ressource partagée
#le répertoire /home/partage doit exister dans l'arborescence linux
path=/home/partage
browseable = yes
writable = yes
create mask = 0777
directory mask = 0777
```

Partage de fichiers

- **Fichier de configuration d'un serveur SAMBA :**

- **Configuration du partage de répertoires**
  - Notons que le fichier smb.conf contient déjà des options par défaut lors de l'installation.
  - il faut seulement modifier les options si besoin créer les répertoires partagés.
  - Voici ci-après un exemple de fichier de configuration.

Partage de fichiers

- **Fichier de configuration d'un serveur SAMBA :**

```
workgroup = MON_WORKGROUP
server string = %h server (Samba %v) // %v version samba %h nom du serveur samba
dns proxy = no
security = user //nécessite une authentification pour accéder aux ressources
encrypt passwords = true
socket options = TCP_NODELAY
log file = /var/log/samba/log.%m // %m: nom de la machine cliente
[homes]
comment = Home Directories
browseable = no
writable = yes
create mask = 0775
directory mask = 0775
valid users = toto, webmaster
# déclaration d'un répertoire partagé "partage"
# ouvert en écriture dans /home/partage
[partage]
comment = Repertoire partagé
browseable = no
writeable = nes
public = no
path = /home/partage
```

Partage de fichiers

- **Fichier de configuration d'un serveur SAMBA :**
  - Avec l'option **`public = no`** ou les options **`valid users =,`** les utilisateurs doivent s'authentifier lors de la connexion aux ressources partagées.
  - Pour créer un compte samba pour l'utilisateur toto:

\# smbpasswd -a MonCompte MonMotdePasse

Cette commande ajoute le compte SAMBA MonCompte avec le mot de passe MonMotDePasse.

  - Il est possible ensuite dans la section "Share définitions" d'ajouter des partages accessibles seulement à certains utilisateurs. (voir l'exemple ci-après) par exemple pour le répertoire /home/administration

Partage de fichiers

- **Fichier de configuration d'un serveur SAMBA :**

- par exemple pour le répertoire /home/administration

```
[administration]
path=/home/administration
public = no
valid users = pierre @admin
writable = yes
create mask = 0770
```

Le paramètre @admin permet de donner des droits aux membres du groupe système ***admin****.*

Le répertoire /home/administration doit être créé sous linux avec les droits adéquats , par exemple:

```
mkdir /home/administration
chown pierre:admin /home/administration
chmod 770 /home/administration
```

Partage de fichiers

- **Lancement du serveur samba**

  - Pour ubuntu 9 et moins:

  ➢ lancer *samba* :

```
/etc/init.d/samba start
```

  ➢ arrêter *samba* :

```
/etc/init.d/samba stop
```

  ➢ Redémarrer proprement *samba* après un changement de configuration dans **smb.conf** :

```
/etc/init.d/samba restart
```

  - Pour ubuntu 10 et plus, on utilise **smbd** au lieu de **samba**

```
service smbd start|stop|restart
```

Partage de fichiers

- **Accès depuis un poste client**

- **Accès aux répertoires partagés à partir de linux :**

  pour visualiser tous les partages d'un serveur :

  ```
  smbclient -L //@ip-du-serveur -U NomdeCompte
  ```

  Entrez ensuite le mot de passe Samba du compte

  Vous pouvez également mettre le nom du serveur à la place de l'adresse IP du serveur.

  pour visualiser un répertoire partagé du serveur :

  ```
  smbclient //nom_du_serveur/nom_du_répertoire -U
  nom_utilisateur -I IP_du_server
  ```

  *Dans ce cas, **smbclient** se comporte comme un client ftp, avec les commandes **get**, **put**, etc...*

Partage de fichiers

- **Accès depuis un poste client**

- **Accès aux répertoires partagés à partir de Windows:**

  Dans l'interface de *Windows XP*, accédez au voisinage réseau par :

  - Favoris réseaux ;
  - Voir les ordinateurs du groupe de travail ;
  - Réseau Microsoft Windows

  Seuls les répertoires avec l'option `browsable = yes` apparaissent dans l'interface de l'explorateur *Windows*.

  En cliquant sur le serveur, vous devrez entrer un login et un mot de passe valides pour accéder aux partages du serveur SAMBA

Partage de fichiers

- **Serveur Samba en tant que contrôleur de domaine:**
  - Pour devenir contrôleur de domaine, c'est à dire l'équivalent d'un serveur Windows, il faut tout d'abord modifier la section globale du fichier **smb.conf**.

```
[global]

   workgroup = mondomaine      //à remplacer par votre nom de domaine
   netbios name = nom_serveur  // à remplacer par votre nom de serveur

# paramètres samba
   logon drive = U:
   logon home = \\serveur\%U
   logon path = \\serveur\profiles\%U
   logon script = logon.cmd  //ce script devra être créé et stocké dans le
répertoire Netlogon


   # Controle de domaine
   os level = 65
   domain logons = yes
   domain master = yes
   local master = yes
   preferred master = yes
   wins support = yes

# Base de comptes
   passdb backend = tdbsam:/var/lib/samba/mypassdb.tdb, guest
# le reste de cette section est le même que dans le premier exemple
```

Partage de fichiers

- **Serveur Samba en tant que contrôleur de domaine:**
- Il faut créer ensuite des partages spécifiques "homes" qui contiendra les répertoires spécifiques des utilisateurs:

```
#====================== Share Definitions ========================

[homes]
   path = /data/samba/home/%u
   comment = Home Directories
   valid users = %S
   guest ok = no
   browseable = no
   writable = yes
   create mask = 0700
   directory mask = 0700
```

Partage de fichiers

- **Serveur Samba en tant que contrôleur de domaine:**
- Il faut créer ensuite "**netlogon**" qui contiendra les scripts de connexion exécutés par les machines clientes à chaque connexion d'un utilisateur,

```
#====================== Share Definitions ======================

[homes]
   ...
[netlogon]
    comment = Partage Netlogon
    path = /data/samba/netlogon
    guestok = no
    readonly = yes
    browseable = no
    writable = no
    valid users = @sambausers
    create mask = 0777
    directory mask = 0777
```

Partage de fichiers

- **Serveur Samba en tant que contrôleur de domaine:**
- "profiles" qui permet de stocker de manière centralisée la configuration du bureau etc ... de chaque utilisateur du domaine.

```
#====================== Share Definitions ======================

[homes]
…..
[netlogon]
…..
[profiles]
    path = /data/samba/profiles
    comment = Repertoires Profiles
    guest ok = no
    writable = yes
    create mode = 0700
    browsable = no
    valid users = @sambausers
# le reste de cette section est le même que dans le premier exemple
```

Partage de fichiers

- **Serveur Samba en tant que contrôleur de domaine:**
- Créer un groupe ***sambausers*** **avec le RID 513** pour être en conformité avec la terminologie Windows. **#groupadd -g 513 sambausers** // (RID=GID=513).
- Ajouter dans ce groupe les utilisateurs linux du domaine. Il faut également créer des comptes Samba pour chacun de ces utilisateurs grâce à la commande **smbpasswd** vue précédemment.
- Ne pas oublier d'ajouter le compte "root" au groupe "sambausers", celui-ci sera le super-utilisateur samba. **#smbpasswd -a root MotDePasse**
- Créer le groupe ***sambamachines*** avec le RID 515

  #groupadd -g 515 sambamachines

Partage de fichiers

- **Serveur Samba en tant que contrôleur de domaine:**
- Les comptes machines du domaine doivent être créés avec un **$** à la fin du nom **netbios**, voici un extrait du fichier "/**etc/passwd**" avec comme exemple 2 machines **clientxp1** et **clientxp2** :

  client1$:x:1003:515::/dev/null:/dev/null

  client2$:x:1004:515::/dev/null:/dev/null
- Créez également un compte Samba pour chaque machine:

  #smbpasswd -a -m NomMachine (sans le $)
  - Création des 3 nouveaux répertoires partagés et attribution des droits:

  #mkdir -p /data/samba/home;chown root:sambausers /data/samba/home

  #mkdir -p /data/samba/netlogon;chown root:sambausers /data/samba/home

  #mkdir -p /data/samba/profiles;chown root:sambausers /data/samba/home

Partage de fichiers

- **Serveur Samba en tant que contrôleur de domaine:**
- Extrait du fichier /etc/group

  sambausers:x:513:**root,smbuser**
  sambamachines:x:515:**clientxp1$,clienxp2$**
- Exemple de script exécuté au démarrage logon.cmd
  - **Exemple de script exécuté au démarrage logon.cmd**

```
@echo off
NET USE H: \\NomServeur\Nompartage
@echo on
```

  Pour démarrer et arrêter le serveur la commande reste la même

  sudo /etc/init.d/samba Start|Stop|Restart
- Vous pouvez enfin vérifier que le partage est monté sur les machines clienxp1 et clientxp2 .

Troisième partie

Applications FTP, Telnet, SFTP et SSH.

Telnet et FTP

- **Présentation des concepts importants**

- **Telnet:** Telnet est un protocole qui permet l'émulation de terminal VTx à distance sur un serveur Unix/Linux.

  FTP: FTP est un protocole de communication qui permet le transfert de fichiers entre plusieurs machines.

- Sous linux, les serveurs les plus couramment utilisé est *wu-ftpd* et telnetd. Or l'inconvénient de ces serveurs c'est qu'ils sont très peu sécurisés et laissent passer les mots de passe en clair.
- Il existe d'autres serveurs tel que *Proftpd, stelnetd* qui sont des serveurs demandant moins de ressources et offrant une sécurité accrue, bien qu'elle soit loin de *ssh* et *sftp*.

Telnet et FTP

- **Présentation des concepts importants**

- **Le daemon inetd:** Le daemon INETD est un « super » serveur, à l'écoute sur plusieurs ports et qui se charge de recevoir les demandes de connexion de plusieurs clients (telnet, ftp,...) et de lancer le serveur correspondant à la demande. A son démarrage il consulte les fichiers:
  - */etc/services* qui contient la liste générale des services TCP/IP avec leur numéro de port et le protocole de transport associé.
  - */etc/inetd.conf* qui contient la liste des services activés sur une machine donnée
- Dans les distributions linux récentes, le daemon **inetd** a été remplacé par **xinetd**.

Telnet et FTP

- **Extrait de /etc/services et /etc/inetd.conf:**

- */etc/services*

```
ftp  21/tcp
telnet  23/tcp
smtp  25/tcp  mail
pop3  110/tcp  # Post Office
```

- /etc/inetd.conf

```
ftp      stream  tcp     nowait  root    /usr/sbin/ftpd          ftpd
#shell   stream  tcp     nowait  root    /usr/sbin/rshd          rshd
#login   stream  tcp     nowait  root    /usr/sbin/rlogind       rlogind
#exec    stream  tcp     nowait  root    /usr/sbin/rexecd        rexecd
```

Ici, il n'y a que le service ftp qui est activé par le serveur **inetd**. Les autres lignes sont en commentaires. Ces services sont dits fonctionnant en mode « parallèle ».

Telnet et FTP

- **Configuration avec xinetd:**

Dans les distributions linux récentes, *inetd* a été remplacé par ***xinetd***. à la différence que vous avez un fichier de configuration global ***"/etc/xinetd.conf"***, et un fichier de configuration par service dans ***/etc/xinetd.d***.

```
.....
# Le fichier xinetd.conf
.....
defaults
{
        instances               = 60
        log_type                = SYSLOG authpriv
        log_on_success          = HOST PID
        log_on_failure          = HOST
        cps                     = 25 30
}

includedir /etc/xinetd.d
```

Telnet et FTP

- **Configuration avec xinetd:**

- ***instances*** - Détermine le nombre maximal de requêtes qu'un service xinetd peut gérer à un moment donné.
- ***log_type*** - Configure xinetd de sorte qu'il utilise la facility de journalisation authpriv qui enregistre des entrées de journalisation dans le fichier /var/log/secure. L'ajout d'un directive telle que FILE /var/log/xinetdlog entraînerait la création d'un fichier de journalisation personnalisé portant le nom xinetdlog dans le répertoire /var/log/.
- ***log_on_success*** - Configure xinetd de façon à ce qu'il effectue la journalisation si la connexion est établie avec succès.
- ***log_on_failure*** - Configure xinetd de façon à ce qu'il effectue la journalisation si la connexion échoue ou si elle n'est pas autorisée.
- ***cps*** — Configure xinetd de manière à n'autoriser que 25 connexions par seconde à un service donné. Si cette limite est atteinte, le service est retiré pendant 30 secondes.
- ***includedir*** /etc/xinetd.d/ - Inclut des options stipulées dans les fichiers de configuration spécifiques aux services qui se trouvent dans le répertoire /etc/xinetd.d/.

Telnet et FTP

- **Répertoire /etc/xinetd.d/**
  - **Le fichier /etc/xinetd.d/ftp:**

```
.....
service ftp
{
        disable                 = no
        socket_type             = stream
        wait                    = no
        user                    = root
        server                  = /usr/sbin/in.ftpd
        server_args             = -l -a
        log_on_success          += DURATION USERID
        log_on_failure          += USERID
        nice                     = 10
}
```

Telnet et FTP

- **Répertoire /etc/xinetd.d/**
  - **Le fichier /etc/xinetd.d/telnet:**

```
.....
service telnet
{
        flags                   = REUSE
        socket_type             = stream
        wait                    = no
        user                    = root
        server                  = /usr/sbin/in.telnetd
        log_on_failure          += USERID
        disable                 = yes
}
```

Telnet et FTP

- **TCP-Wrapper:**
- TCP-Wrapper est un outil de sécurité réseau qui permet de contrôler les accès, les tentatives de connexion sur une machine donnée.
- Il permet à tout instant de savoir (par journalisation syslogd) qui essaie d'accéder sur un ordinateur mais également de filtrer les accès.
  - On peut par exemple sur une machine A interdire les connexions telnet venant d'une machine B tout en autorisant les connexions FTP venant de cette même machine B.
- Tcpwrapper sert d'enveloppe. Il vient « **s'intercaler** » entre le daemon **inetd** et le serveur à démarrer. Quand une demande de service TCP/IP (en réalité TCP ou UDP) arrive sur un port donné, *inetd* va lancer *TCPD* (daemon correspondant à Tcpwrapper) au lieu d'activer directement le service demandé (telnetd, ftpd, pop3...).

Telnet et FTP

- **Eléments de configuration:**
- Sous Linux, **tcpd** est installé par défaut. On peut voir en consultant le fichier /etc/inetd.conf comment **inetd** active **tcpd**.

➢ **Extrait de /etc/inetd.conf**

```
ftp      stream  tcp     nowait  root    /usr/sbin/tcpd  in.ftpd -l -a
telnet   stream  tcp     nowait  root    /usr/sbin/tcpd  in.telnetd
```

➢ **TCP Wrapper**

- L'administrateur réseau va pouvoir utiliser 2 fichiers: /etc/hosts.allow et /etc/hosts.deny pour filtrer les accès à sa machine.
  - *<u>/etc/hosts.deny</u>*: on indique dans ce fichier les services et les hôtes pour lesquels l'accès est interdit.
  - *<u>/etc/hosts.allow</u>***:** on indique dans ce fichier les services et les hôtes pour lesquels l'accès est autorisé.

Telnet et FTP

- **Eléments de configuration:**

➢ **TCP Wrapper** (Exemple):

```
# Fichier /etc/hosts.deny
# interdit tous les accès ftp à la machine
in.ftpd:ALL

# Fichier /etc/hosts.allow
# autorise les accès ftp venant de cli1
in.ftpd :cli1.archinet.edu
```

TCP-Wrapper utilise l'algorithme suivant :

Si une règle est applicable dans hosts.allow, alors cette règle est appliquée, sinon, Si une règle est applicable dans hosts.deny alors cette règle est appliquée, sinon, l'accès est autorisé.

Telnet et FTP

- **Eléments de configuration:**

➢ **TCP Wrapper** (Exemple):

- Le mode de fonctionnement de TCPWrapper induit la stratégie de sécurité à adopter :
  1. décrire toutes les règles pour les couples (services/clients) qui sont autorisés,
  2. interdire systématiquement tout le reste. Mettre par défaut ALL:ALL dans hosts.deny.

- Les tentatives d'accès depuis des machines extérieures sont toutes enregistrées dans des fichiers particuliers. Ces enregistrements sont effectués par le processus **syslogd** qui, à son démarrage, lit le fichier **/etc/syslog.conf** pour trouver dans quel(s) fichier(s) il doit enregistrer les différentes tentatives d'accès.

Telnet et FTP

- **Extrait de /etc/syslog.conf:**

```
# Log anything (except mail) of level info or higher.
# Don't log private authentication messages

# The authpriv file has restricted access.
authpriv.*                                              /var/log/auth.log
auth,authpriv.none;                                     /var/log/syslog
```

- **Extrait de /var/log/syslog**

```
Feb  3 18:02:52 ns1 ftpd[1051]: FTP session closed
Feb  3 18:03:31 ns1 syslogd 1.3-3: restart.
Feb  3 18:07:34 ns1 in.ftpd[1057]: refused connect from
cli1.archinet.edu
Feb  3 18:07:46 ns1 in.ftpd[1058]: connect from ns1.archinet.edu
Feb  3 18:10:57 ns1 login[1063]: LOGIN ON ttyp3 BY mlx FROM puce
```

Telnet et FTP

- **Lancement des services FTP et Telnet**
- Pour démarrer les services Telnet et FTP lancer la commande :

```
sudo /etc/init.d/xinetd start|stop|restart
```

Ou

```
Sudo service xinetd start|stop|restart
```

- **Accès depuis un client**

Pour FTP:

```
ftp @IP(serveur) ou ftp FQDN(serveur FTP)
.....
ftp> help # pour avoir les commandes possibles
```

Pour Telnet:

```
telnet @IP(serveur) ou telnet FQDN(serveur Telnet)
....
farid@Serveur:~$
```

scp, sftp et les tunnels avec ssh

- **Présentation**
- Il est possible de récupérer sans difficulté les mots de passe des personnes utilisant le réseau si ces derniers ne sont pas cryptées et transitent en clair sur les réseau.
- Il existe des solutions permettant de sécuriser un minimum les transactions.
  - **SSH** ou **Secure SHell**, propose un shell sécurisé pour les connexions à distance permettent ainsi d'établir un tunnel crypté (ou chiffré) entre le client et le serveur.
  - "**scp**" ou "**sftp**" sont des programmes de la même famille remplacent les commandes ftp ou rcp.
- SSH sur GNU/Linux est généralement composé de 3 packages :
  - **OpenSSH** général**, (openssh),** requièrent le paquetage **OpenSSL (openssl).**
  - le serveur **OpenSSH (openssh-server)**
  - le client **(openssh-clients).**

scp, sftp et les tunnels avec ssh

- **Mode de fonctionnement de SSH**
- L'établissement du dialogue entre le client et le serveur suit un protocole particulier :
  1. établissement d'une couche transport sécurisée
  2. chiffrement des données à l'aide de clefs symétriques pendant la transaction

  Le client peut s'authentifier en toute sécurité, et accéder aux applications conformes aux spécifications du protocole.

scp, sftp et les tunnels avec ssh

- **Mode de fonctionnement SSH (couche transport).**
- La couche transport assure le chiffrement et le déchiffrement des données.

Elle assure également la compression pour améliorer le transfert. Le client et le serveur négocient plusieurs éléments afin que la session puisse s'établir.

1. l'échange des clés
2. l'algorithme de clé publique à utiliser
3. l'algorithme de chiffrement symétrique à utiliser
4. l'algorithme d'authentification de message à utiliser
5. l'algorithme repère (hash) à utiliser

- Lors du premier échange, le client ne connaît pas le serveur. Le serveur propose alors une clé hôte qui servira par la suite au client de moyen d'identification du serveur.

scp, sftp et les tunnels avec ssh

- **Mode de fonctionnement SSH**
- **Authentification:** Une fois le tunnel sécurisé mis en place, le serveur envoie au client les différentes méthodes d'authentification qu'il supporte. Dans le cadre d'une authentification par mot de passe, celui-ci peut être envoyé en toute sécurité puisqu'il est chiffré.
- **Connexion:** Une fois l'authentification réalisée, le tunnel SSH peut multiplexer plusieurs canaux en délégant la tâche à des agents
- **Fichiers de configuration d'OpenSSH:** OpenSSH est constitué de deux ensembles de fichiers de configuration: Il y a un fichier de configuration pour les programmes clients (**ssh, scp** et **sftp**) et l'autre pour le service serveur (***sshd***).

scp, sftp et les tunnels avec ssh

- **Fichiers de configuration d'OpenSSH:**
- Voici les principaux fichiers de configuration :

```
ssh_config              fichier de configuration client SSH pour l'ensemble
                        du système. Il est écrasé si un même fichier est
                        présent dans le répertoire personnel de
                        l'utilisateur (~/.ssh/config).
sshd_config             fichier de configuration pour sshd.
ssh_host_dsa_key        clé DSA privée utilisée par sshd.
ssh_host_dsa_key.pub    clé DSA publique utilisée par sshd.
ssh_host_key            clé RSA privée utilisée par sshd pour la
                        version 1 du protocole SSH.
ssh_host_key.pub        clé RSA publique utilisée par sshd pour la
                        version 1 du protocole SSH.
ssh_host_rsa_key        clé RSA privée utilisée par sshd pour la
                        version 2 du protocole SSH.
ssh_host_rsa_key.pub    clé RSA publique utilisée par sshd pour la
                        version 2 du protocole SSH.
```

scp, sftp et les tunnels avec ssh

- **Fichiers de configuration d'OpenSSH:**
- Les informations spécifiques à un utilisateur sont stockées dans son répertoire personnel à l'intérieur du répertoire *~/.ssh* :

```
authorized_keys ou parfois authorized_keys2
                        ce fichier contient une liste de clés publiques "autorisées". Si
                        un utilisateur se connecte et prouve qu'il connaît la clé privée
                        correspondant à l'une de ces clés,  il obtient
                        l'authentification. Notez qu'il ne s'agit que d'une méthode
                        d'authentification facultative.
id_dsa                  contient l'identité d'authentification
                        DSA de l'utilisateur.
id_dsa.pub              la clé DSA publique de l'utilisateur.
id_rsa                  la clé RSA publique utilisée par sshd pour
                        la version 2 du protocole SSH.
identity                la clé RSA privée utilisée par sshd pour la
                        version 1 du protocole SSH.
known_hosts             ce fichier contient les clés hôte DSA des serveurs SSH auxquels
                        l'utilisateur s'est  connecté.
```

scp, sftp et les tunnels avec ssh

- **Configurer et utiliser SSH:**
- vous devez mettre en place les clés qui serviront à ssh pour vous authentifier. Concrètement cela consiste à définir une paire de clés, une publique que vous mettrez sur le serveur distant, une privée que vous conserverez sur votre machine.
- La première chose à faire est de vous créer une clé. Voyons comment réaliser cela.
  - Allez dans votre répertoire personnel (commande `cd`)
  - Lancer la commande **ssh-keygen -t dsa.** Cette commande permet de générer une clé DSA par défaut de 1024 bits. La clé privée sera stockée dans **~/.ssh/id_dsa** et la clé publique dans **~/.ssh/id_dsa.pub**.

scp, sftp et les tunnels avec ssh

- **Configurer et utiliser SSH:**
- Si vous voulez générer une clé **RSA2**, utilisez l'option "`-t rsa`" et pour du RSA1 "`-t rsa1`".
- Vous devrez entrer une "*passphrase*". Entre 10 et 30 caractères. Mélangez majuscules, minuscules et chiffres. La clé privée doit ensuite être mise en lecture seule pour le propriétaire et aucun accès pour les autres.
- Pour modifier votre "passphrase" sur une clé privée DSA, utilisez la commande : `ssh-keygen -p -f ~/.ssh/id_dsa`.
- La **clé publique**, doit être **copiée sur le serveur distant** dans *~/.ssh/authorized_keys*. La clé privée reste sur votre poste client.

scp, sftp et les tunnels avec ssh

- **Configurer et utiliser SSH:**
- Vous pouvez mettre plusieurs clés publiques sur le serveur, si vous le souhaitez ou si vous accédez au serveur avec plusieurs comptes d'accès différents.
- Copiez la clé avec scp sur le compte que vous avez sur le serveur :

```
$ cat .ssh/id_dsa.pub | ssh mlx@M1.foo.org \
"cat - >>.ssh/authorized_keys"
```

- Le système demande votre mot de passe. Une fois saisi, les prochaines connexion **ssh** se feront qu' après saisi de la *passphrase,* chose qui est encore fastidieux.

  Pour essayer de se passer de ça, nous utilisons un **agent ssh.**

scp, sftp et les tunnels avec ssh

- **Configurer et utiliser SSH:**

  Remarque : Envoyer une clé par mail n'est pas un système sûr, et même chiffré et signé cela ne garantit pas au destinataire que vous en êtes l'émetteur s'il ne vous a jamais vu.

  L'administrateur distant peut demander à ce que l'envoyeur justifie qu'il est bien celui qui a envoyé la clé. Il suffit pour cela de téléphoner à l'administrateur et de communiquer **"la signature ou empreinte" (finger print)** de la clé (ou par sms).

  Pour obtenir le "**finger print**" d'une clé utiliser la commande :

  $ `ssh-keygen -l`

scp, sftp et les tunnels avec ssh

- **Configurer et utiliser SSH:**
- **Utiliser un agent ssh:** L'utilisation d'un agent, évite d'avoir à retaper la "passphrase" à chaque fois que l'on sollicite l'utilisation de la clé privée. Un agent stocke en mémoire les clés privées. Voici comment activer un agent :
  1. Lancer la commande `ssh-agent`
  2. La commande met sur la sortie standard des variables environnement à déclarer et à exporter. Faites le.
  3. exportez les clés. Cela consiste à les mettre dans le cache de l'agent avec la commande `ssh-add`. La commande demandera la "passphrase"..

Pour supprimer une clé (ici RSA) de l'agent, utilisez l'option "-d"

```
$ ssh-add -d ~/.ssh/id_rsa
```

scp, sftp et les tunnels avec ssh

- **Utilisation de SCP et de SFTP:**
- L'utilisation de ces commandes est relativement simple. SCP permet de faire de la copie de fichiers. SFTP est utilisable en mode interactif ou en mode batch et ressemble plus au FTP.
- **Utilisation de scp** (exemple d'utilisation):

```
farid@Client1:~$ ssh farid@serveur "ls -al tmp"
total 33832
-rw-r--r-- 1 farid farid 25357824 2013-03-12 07:20 openldap-2.4.34.tar
-rw-r--r-- 1 farid farid  9281894 2013-03-12 05:18 db-4.5.20.tar.gz
```

Cette commande donne la liste des fichiers distants qui sont dans le répertoire "~/tmp". Pour copier ces fichiers localement dans un répertoire tmp :

```
farid@Client1:~$ cd && mkdir tmp ;scp farid@serveur:/home/farid/tmp/* ~/tmp
db-4.5.20.tar.gz                              100% 9064KB   4.4MB/s   00:02
openldap-2.4.34.tar                           100%   24MB   4.8MB/s   00:05
```

scp, sftp et les tunnels avec ssh

- **Utilisation de SCP et de SFTP:**
- **Utilisation de SFTP** (exemple d'utilisation): sftp peut être utilisé pour du transfert de fichiers en mode sécurisé.

```
farid@Client1:~$ sftp serveur
Connected to serveur.
sftp>
```

On obtient un prompt, ici le système ne m'a pas demandé de m'authentifier

Pour avoir une liste des commandes, utiliser "help".

```
sftp> help
Available commands:
bye                                Quit sftp
cd path                            Change remote directory to 'path'
chgrp grp path                     Change group of file 'path' to 'grp'
chmod mode path                    Change permissions of file 'path' to 'mode'
chown own path                     Change owner of file 'path' to 'own'
df [-hi] [path]                    Display statistics for current directory or
                                   filesystem containing 'path'
exit                               Quit sftp
```

FIG: Extrait des commandes utilisé en SFTP

scp, sftp et les tunnels avec ssh

- **Tunnels SSH: Relayage de ports avec ssh**
- SSH permet aussi de fournir un moyen simple pour créer un tunnel parfaitement sûr. Les deux options dont nous détaillerons les effets sont -L (port Local) et -R (port distant - *Remote*).
- Le relayage de ports (*port forwarding*) est souvent troublant car de nombreux éléments entrent en compte : **un client et un serveur ssh**, **un client et un serveur TCP** dont la connexion sera encapsulée dans le trafic ssh, soit quatre intervenants.
- La différence entre **relayage local** et **distant** vient du « sens de la connexion ». Le client ssh se connecte sur le serveur ssh. En revanche, selon l'emplacement du client et du serveur TCP par rapport au client et au serveur ssh, le relayage est local ou distant
  - **local** : le client TCP et le client ssh sont du même côté
  - **distant** : le serveur TCP et le client ssh sont du même côté.

scp, sftp et les tunnels avec ssh

- **Tunnels SSH: Relayage de ports avec ssh**
- **Redirection locale**
  - Synatxe:

ssh -L <port-local-à-relayer>:<machine-distante>:<port-distant> machine-distante

  - Exemple:

client1$ ssh –L 1234:<hostname>:21 serveur

- Cette commande ouvre une connexion sur ***client1*** vers ***serveur***. En fait, un tunnel est ouvert depuis le *port 1234* de **client1**, à destination du *port 21* de ***hostname***. Ce tunnel part de ***localhost:1234*** vers s***erveur:22*** pour terminer en ***hostname:21*.**
- Après cette commande, une connexion sur le port 1234 de client1 est donc dirigées vers différents endroits, en fonction de qui est ***hostname*.**
- ***hostname*** peut être ***localhost***, ***serveur*** ou une autre machine (***client2***)

scp, sftp et les tunnels avec ssh

- **Tunnels SSH: Relayage de ports avec ssh**
- **Redirection locale:** (suite exemple)
  - prenons le cas ou ***hostname*** est ***client2***

    client1$ ssh –L 1234:client2:21 serveur

  Une connexion est ouverte entre ***client1*** et ***serveur***, mais le relayage de ports n'est pas entre ces deux hôtes, mais entre ***client1:1234*** et ***client2:21.***

  Attention, dans ce cas le trafic entre serveur et client2 n'est pas chiffré !!! Lancer la commande ***tcpdump –X host client2*** *sur serveur pour vous en rendre compte.*

  Il est possible de chiffrer cette connexion en echaînant les tunnels:

  client1$ ssh –L 1234:localhost:5678 serveur

  …

  serveur$ ssh –L 5678:localhost:21 client2

scp, sftp et les tunnels avec ssh

- **Tunnels SSH: Relayage de ports avec ssh**
- **Redirection distante:**
  - ❖ Synatxe:

  ssh -R <port-distant-à-relayer>:<machine-distante>:<port-local-du-serveur> machine-distante

  - ❖ Exemple:

  client1$ ssh –R 1234:<hostname>:21 serveur

  L'hôte depuis lequel est lancé cette commande sert alors de relais entre ***serveur:1234*** et ***hostname:21***. Cela sert par exemple lorsque l'administrateur de ce relais (client1 dans notre exemple) souhaite donner une autorisation de connexion à une machine externe vers une machine interne à un réseau.

OFPPT

Quatrième partie
Routage, firewall

Routage

- Le routage permet de faire communiquer plusieurs sous-réseaux.
- Une passerelle (en anglais gateway) Assure la communication entre les différents sous-réseaux
- Une route définie sur une station est un chemin que doivent prendre les paquets d'un certain sous-réseau à destination .

Routage

- Soit le schéma suivant:

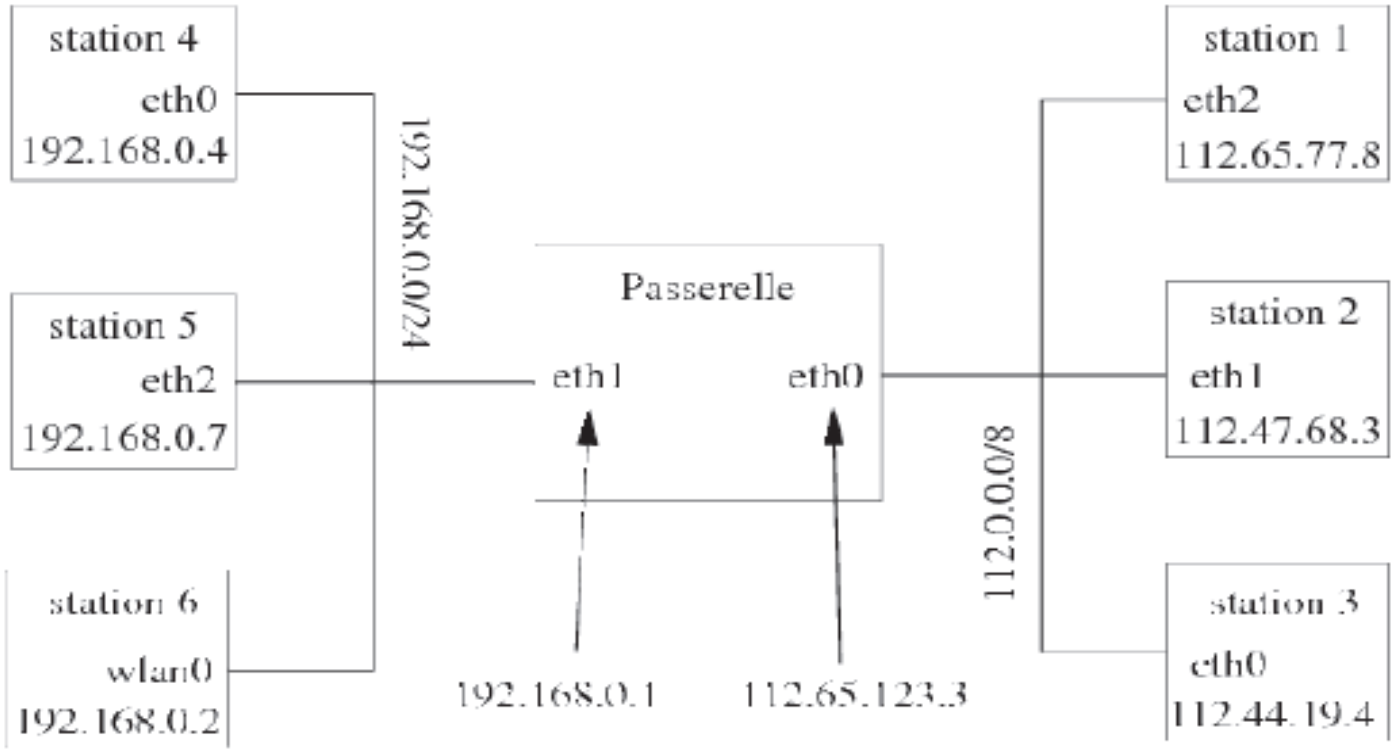


Fig-2: Exemple de passerelle faisant communiquer deux réseaux

Routage

Si la station 1 veut communiquer directement avec *station* 6, trois condition doivent être réunies :

- ✓ Une route au réseau 192.168.0.0/24 doit être définie sur la station 1.

Pour cela, on peut utiliser la commande route :

```
# route add -net 192.168.0.0/24 gw 112.65.123.3
```

Une route au réseau 112.0.0.0/8 doit être définie sur la station 6.

Pour celà, on peut utiliser la commande route :

```
# route add -net 112.0.0.0/8 gw 192.168.0.1
```

La passerelle doit être configurée pour transmettre (ou *forwarder*) les paquets *IP* d'un réseau à l'autre, ce qui se fait par la commande :

```
# echo 1 >/proc/sys/net/ipv4/ip_forward
```

Routage

- Il faut refaire ces commandes après un reboot.
- Pour remédier à ça, vous pouvez mettre vos commandes de routage dans un script d'initialisation au démarrage avec la commande `update-rc.d` (sous *debian*)).
- Pour ajouter un script my_script à l'initialisation:

```
# mv my_script /etc/init.d
# update-rc.d my_script defaults
```

Routage

On peut voir l'état des routes avec la commande route -n. Par exemple, sur la station 1 :

```
# route -n
Destination Gateway Genmask Flags Metric Ref Use Iface
192.168.0.0 112.65.123.3 255.255.255.0 U 0 0 0 eth2
```

etc...

Sur la station 6:

```
# route -n
Destination Gateway Genmask Flags Metric Ref Use Iface
112.0.0.0 192.168.0.1 255.0.0.0 U 0 0 0 wlan0
```

etc...

Pour supprimer une route, par exemple vers le réseau 193.86.46.0/24 via une passerelle 196.24.52.1, on fait :

```
# route del -net 193.86.46.0/24 gw 196.24.52.1
```

Routage

- **Route par défaut (*gateway*)**
  - En général, c'est la route qu'il faut employer pour aller sur internet.
  - Elle sert aussi à acheminer les paquets IP à destination des réseaux non prévus dans les autres routes.

Exemple, pour définir la route par défaut via la passerelle 194.56.87.1 :

```
route add default gw 194.56.87.1
```

Pour supprimer cette même route :

```
route del default gw 194.56.87.1
```

Routage

- **NAT et masquerading**

Un hôte ayant une adresse *IP* sur un réseau local a besoin d'une adresse *IP* sur un autre réseau tel que internet, cet hôte a besoin d'une adresse *IP* sur ce réseau.

Deux solutions:

- soit on demande à ce que les adesses du réseau local soient routées sur le réseau global
- prêter l'*IP* de la passerelle aux machines du réseau local.

➔ Pour cela, on utilise `iptables` avec *NAT*

Routage

- **NAT et masquerading**

Par exemple, si la passerelle se connecte à internet via son interface eth0, il suffit d'exécuter la commande suivante sur la passerelle :

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

On peut aussi donner aux machines du réseau local une autre adresse *IP* que l'on spécifie avec `--to` :

```
iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to
193.56.17.9
```

Protocoles, services, ports

- **La liste des protocoles connus du systèmes**
  - Un protocole (*IP*, *TCP*, *UDP*,...) est un mode de communication réseau, c'est à dire une manière d'établir le contact entre machines et de transférer les données.
  - Sous linux, la liste des protocoles reconnus par le système se trouve dans le fichier /etc/protocols.
  - Exemple: Le protocole **TCP** sert à établir une communication fiable entre deux hôtes au contraire de l'UDP.

Firewall configurés avec *iptables*

- Firewall ou pare-feu configuré avec la commande iptables permet de filtrer les paquets réseau entrants, sortants, et transmis, sur une machine.
- On peut filtrer par interface, par port, par adresse de source ou de destination des paquets.
- iptables permet aussi de partager une adresse *IP* (par exemple sur internet) entre plusieurs machines d'un réseau local.

Firewall configurés avec *iptables*

- **Principe d'iptables (Les politiques)**

• On peut adopter 3 politiques: ACCEPT, REJECT ou DROP.

✓Avec la politique ACCEPT, le paquet est simplement transmis normalement.

✓Avec la politique REJECT le paquet n'est pas transmis et la machine source est prévenue.

✓Avec la politique DROP, le paquet n'est pas transmis et la machine source n'est pas prévenue.

Firewall configurés avec *iptables*

- **Principe d'iptables (Les règles)**
- La configuration iptables consiste en un ensemble de règles.
  - **-A chaine ou -I chaine : La chaîne INPUT, OUTPUT ou FORWARD**
  - **-i interface et/ou -o interface Les interfaces d'entrée et de sortie (optionnel)**
  - **-p protocole Le protocole (si besoin) (voir /etc/protocols)**
  - **--sport ou --dport ; Les ports de source ou de destination (si besoin)**

OFPPT

Firewall configurés avec *iptables*

- **Principe d'iptables (Les règles)**

L'ordre des règles est important.

C'est la différence entre -A et -I. Le -A rajoute la règle à la fin d'une chaîne et le -I au début.

Pour vider toutes les règles :

```
iptables -F chaîne
```

Pour mettre une politique par défaut (qui s'applique à tous les paquets sauf règle contraire) :

iptables -P chaîne politique

Cinquième partie
Protocole LDAP

Protocole d'annuaire *LDAP*

- **LDAP** (Lightweight Directory Access Protocol) est le protocole d'annuaire sur TCP/IP.
- Un annuaire est une base de données pouvant représenter des personnes ou des éléments d'infrastructure réseau ou autre.

But: Accès rapide à l'information des attributs des objets en lecture via des requêtes.

- Les accès en écriture sont moins optimisés. Pour celà, les données de l'annuaire sont organisées de manière arborescente.
- Les données de l'annuaire sont stockées dans un arbre appelé *Directory Information Tree (DIT)*.

Protocole d'annuaire *LDAP*

- **Le Directory Information Tree (DIT) et les entrées**

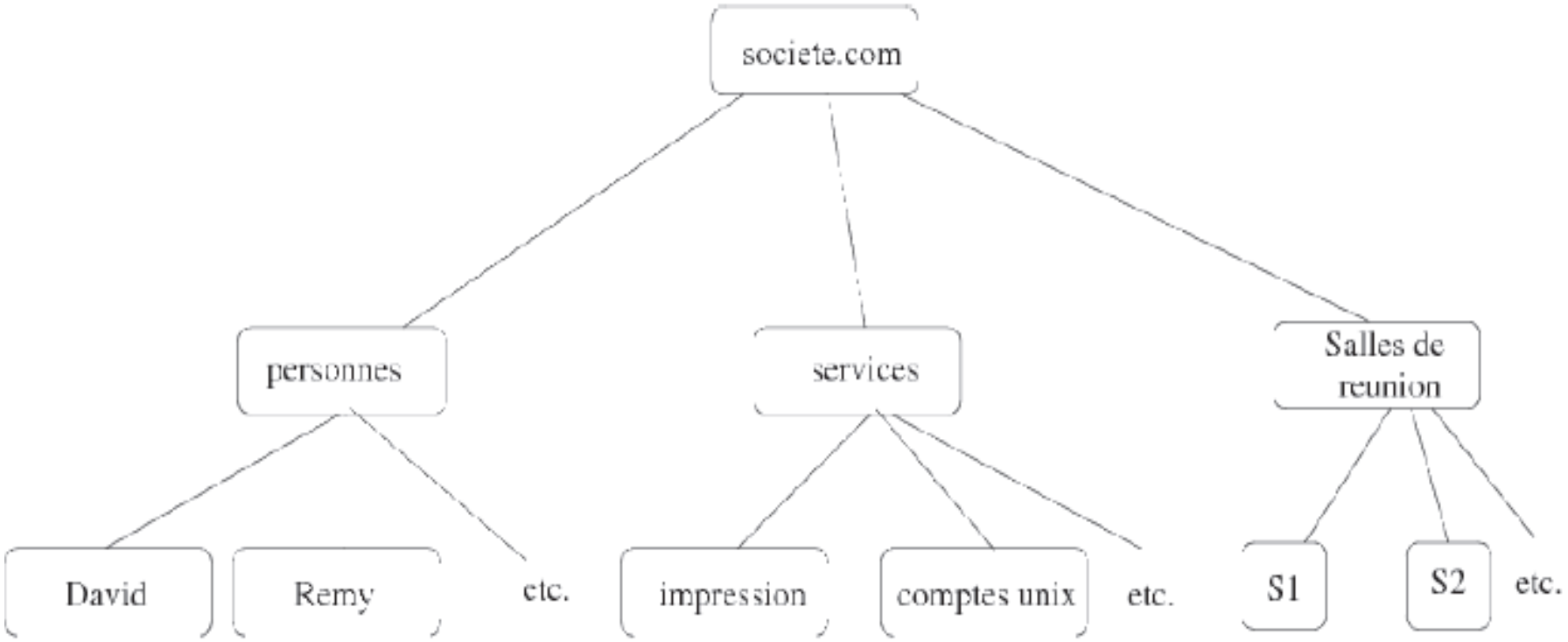


Fig-3 Exemple de Directory Information Tree (DIT)

Protocole d'annuaire *LDAP*

- **Le Directory Information Tree (DIT) et les entrées**

- Chaque noeud est une entrée de l'annuaire (Directory Service Entry, DSE).
- Au sommet de l'arbre se trouve Root Entry (ou Suffix ou BaseDN) qui caractérise la base de données.
- Les autres entrées appartiennent à différentes classes (**objectClass**) qui représentent différentes sortes d'objets:
  - des **groupes** (objectClass organizationalUnit)
  - des **personnes** (objectClass person ou inetOrgPerson, uid)
  - ou des **sevices**, etc.

Protocole d'annuaire *LDAP*

- **Le Directory Information Tree (DIT) et les entrées**
  - Chaque objectClass comporte des **attributs** qui caractérisent les différents objets de la classe. Il y a des attributs obligatoires (**must**) et des attributs facultatifs (**may**).

    Par exemple, une personne de l'objectclass **person** doit avoir un **objectclass**, un nom de famille **sn**, nom usuel ou nom complet **cn**, et peut avoir un prénom (**givenname**), une **description**, un **telephonenumber**, un **userpassword**,...
  - Les classes peuvent dériver les unes des autres comme **inetOrgPerson** qui hérite des attributs de **person**.
  - Lorsqu'on crée une nouvelle objectClass, il faut la soumettre à l'Internet **Assigned Numbers Authority (IANA)** pour que la classe soit officiellement référencée dans LDAP.

Protocole d'annuaire *LDAP*

- **Le Distinguished Name (DN)**
  - Le DN d'une entrée est le chemin dans l'arbre jusqu'à cette entrée à partir de la racine (voir figure 4).
  - Il caractérise l'entrée et permet un accès rapide à cette entrée et à ses attributs dans le DIT.
- **Le Relative Distinguished Name (RDN)**
  - Le RDN est un nom unique dans la branche de l'objet (à un même niveau)
  - Un RDN peut être composé d'un ensemble d'attributs. Le RDN est alors dit *multivalué*

Protocole d'annuaire *LDAP*

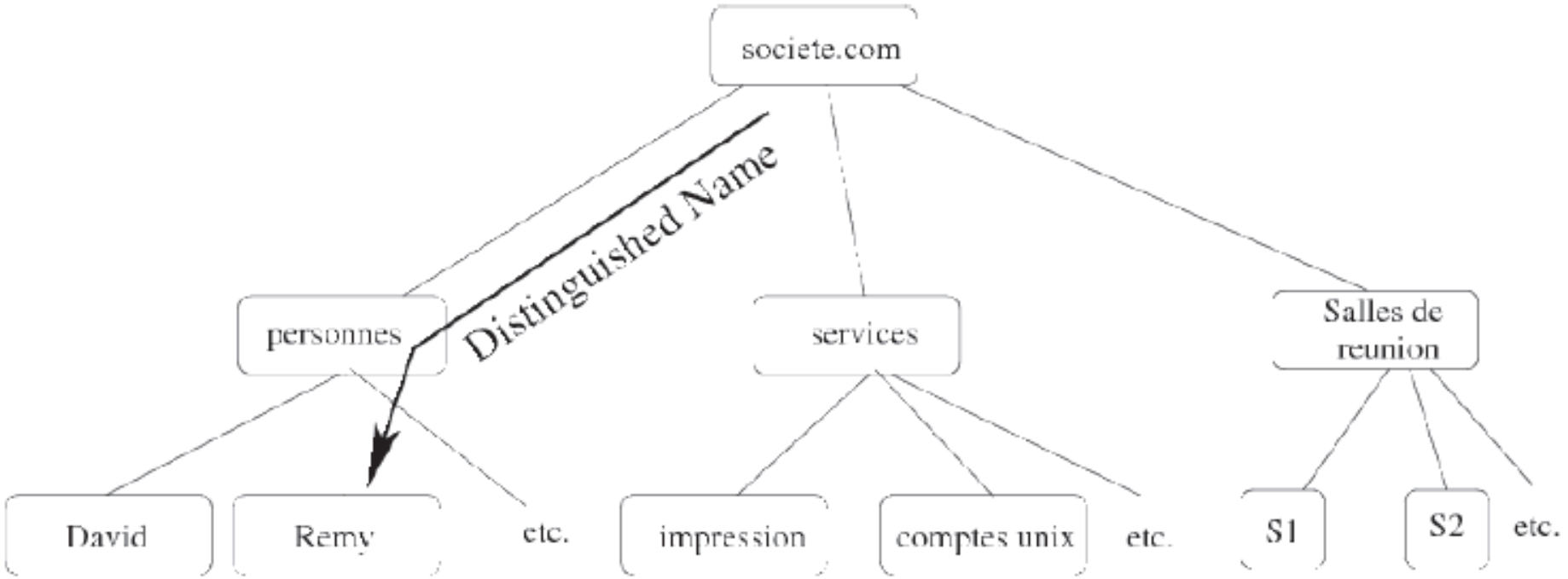


Fig-4: Exemple de Distinguished NameDN

Exemple de DN **dn:cn=Remy, ou=personnes, dc=societe, dc=com.**

Protocole d'annuaire *LDAP*

- **Le modèle Fonctionnel**
  - Le modèle fonctionnel perme d'effectuer des interrogations (recherches dans la base), de comparaison, de mises à jour (création, modification d'entrées), d'authentification (pour droits d'accès) et de contrôle sur le DIT.
  - **Requêtes:** Les requêtes permettent de rechercher des objets et d'afficher leurs attributs suivant différents critères, exemple: **base object, scope, search filter**, **list of attributes** ….

Protocole d'annuaire *LDAP*

- **Le modèle Fonctionnel (requêtes)**
  - **Base object**: l'endroit de l'arbre où doit commencer la recherche
  - **Scope**: la profondeur de la recherche. *Exemple* : **scope = base** (1 seul noeud le base object), **scope=subtree** (tout le sous-arbre sous la base object).
  - **derefAliases** : si on suit les liens vers d'autres BD ou pas
  - **size limit** : nombre de réponses limite (garde-fou)
  - **time limit** : temps maxi alloué pour la recherche (garde-fou)
  - **attrOnly** : renvoie ou pas la valeur des attributs en plus de leur type.
  - **search filter** : le filtre de recherche (requête proprement dite).
  - **list of attributes** : la liste des attributs que l'on souhaite connaître

Protocole d'annuaire *LDAP*

- **Le modèle Fonctionnel**
- **Les filtres de recherche**
  - (sn=Rémy Malgouyres) égalité Nom de famille vaut "Malgouyres"
  - (cn=*elon*) sous-chaîne Nom contient "elon"
  - (cn~=Malgoire) approximation le nom sonne comme "Malgoire"
  - (printedPages>=10000) comparaison printedPages supérieur à 10000
  - (sn=*) existance Tous les noms propres
  - (&(ou=personnes)(telephoneNumber=*)) ET personne et l'attribut numéro de téléphone est connu
  - (|(ou=clermont)(ou=le-puy)) OU ou vaut clermont ou le-puy
  - (!(mail=*)) NON Toutes les entrées sans attribut "mail"
  - (&(objectclass=person)((tel=0473*))) Toutes les entrées de type utilisateur avec numéro de téléphone commençant par 0473.

Protocole d'annuaire *LDAP*

- **Le modèle Fonctionnel**
- **Le client *ldapsearch***
  - Le client **ldapsearch** (**paquet debian ldap-utils**) permet d'effectuer des requêtes en ligne de commande dans un annuaire LDAP.
  - On spécifie les caractéristiques de la requêtes (section précédente) avec le options ci-contre (liste non exhaustive, voir **man ldapsearch** pour plus de détails) :

- **-b 'base object dn'** : spécification du base object (endroit où commencer recherche dans l'arbre) ;
- **-H ldaphost** : spécification du serveur LDAP à consulter ;
- **'filtre'** : filtre à appliquer pour la requête (voir section précedente).
- **atributes** : les attributs des objets à afficher en sortie de la requête.
- **-D 'DN de l'utilisateur'** : connexion en s'identifiant en tant qu'utilisateur via son DN.
- -**W** : prompter pour demander le mot de passe (s'utilise avec -D)
- **-x** : authentification simple non cryptée. Pour une connexion plus sûre, utiliser par exemple SASL.

Protocole d'annuaire *LDAP*

- **Le modèle Fonctionnel**
- **Le client *ldapsearch* (Exemple requête)**
  - Affichage des personnes avec un numéro de téléphone sur le server LDAP example.com :

```
ldapsearch -x -H ldap://example.com -b
  'dc=personnes,dc=societe,dc=com' 'telephoneNumb
```

  - Affichage du nom commun (cn) et de la description de tous les objets de la classe **simpleSecurityObjectConnection** en tant qu'administrateur avec prompt de mot de passe.

```
ldapsearch -x -D 'cn=admin,dc=societe,dc=com' -W -H
  ldap://bigboss -b 'dc=societe, dc=com'
```

Protocole d'annuaire *LDAP*

- **Le modèle Fonctionnel**
  - **Mise à jour de la base**
    - Les opérations de mise à jour permettent de modifier l'état de la base.

      Opérations **add, delete, rename, modify** ;

      - **add** : Ajout d'une entrée qui n'existe pas. Le parent de l'entrée doit avoir été préalablement créé. Les attributs obligatoires de l'entrée (suivant son objectClass) doivent être spécifiés.
      - **rename** : Modification du DN d'une entrée. La nouvelle entrée doit avoir un parent existant. Tout le sous-arbre de cette entrée est déplacé.
      - **modify** : modification du contenu (attributs, objectClass...) d'une entrée.
      - **delete** : suppression d'une entrée. L'entrée ne doit pas avoir d'enfants. Supprimer les sous-arbres récursivement.

Protocole d'annuaire *LDAP*

- **LDIF : LDAP Data Interchange Format**
  - Format de fichier pour faire des **imports/export** d'entrées dans une base ou de bases ou bien faire des modifications sur les entrées.
  - Les fichiers LDIF sont codés en ASCII (norme UTF-8). Toute valeur qui n'est pas en ASCII (nombre, etc..) est codée en ASCII en base 64.
  - Pour importer les entrées d'un fichier LDIF en s'identifiant à partir d'un DN de la base (ici le DN "cn=admin, dc=societe, dc=com")

```
ldapadd -x -D 'cn=admin, dc=societe, dc=com' -f mon_fichier.ldif -W
```

- La forme générale est :

```
dn: <distinguished name>
objectClass: <object class>
objectClass: <object class>
[...]
attribute type:<attribute value>
attribute type:<attribute value>
[...]
```

Exemple de création d'**Organisational Unit** :

```
dn: ou=personnes,dc=societe,dc=com
objectClass: organizationalUnit
ou: personnes
```

- Ci-après un exemple d'entrée de type personne (voir figure 4)

Protocole d'annuaire *LDAP*

dn: cn=Rémy Malgouyres, ou=personnes, dc=societe, dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
uid: remalgou
cn: Rémy Malgouyres
sn: Malgouyres
givenName: Rémy
mail: remy.malgouyres@societe.com
telephoneNumber: 04 73 17 70 00
userPassword: {sha}GBKC57D5LE

dn: cn=David Delon, ou=personnes, dc=societe, dc=com
objectClass: top
objectClass: person
objectClass: organizationalPersonobjectClass: inetOrgPerson
uid: dadelon
cn: David Delon
sn: Delon
givenName: David
mail: david.delon@societe.com
userPassword: {sha}FDHS5J34AH

Protocole d'annuaire *LDAP*

- **LDIF : LDAP Data Interchange Format**
  - La forme générale d'une mise à jour en mode commande est :

```
dn: distinguished name
changetype identifier
change operation identifier
list of attributes...
-
change operation identifier
list of attribute
```

Exemple:

```
dn: cn=Malgouyres, ou=personnes, dc=societe,
dc=com
changetype: modify
replace: telephonenumber
telephonenumber: 04 73 17 70 00
-
add: mobile
mobilenumber = 06 07 08 09 10
```

Protocole d'annuaire *LDAP*

- **Accès à la base à partir d'un client internet**

  syntaxe :

  ldap[s]://<hostname>:<port>/<base_dn>?<attributes>?<scope>?<filter>

  exemples :

  Toute le sous arbre de ou=personnes,dc=societe,dc=com :

  `ldap://ldap.societe.com/ou=personnes,dc=societe,dc=com`

  Telephone number de toutes les personnes de nom de famille

  "Malgouyres"

  `ldap://ldap.societe.com/sn=Malgouyres,ou=personnes,dc=societe,dc=com?telephonenumber`

  Adresses mail de tous les utilisateurs de gmail

  `ldap://ldap.societe.com/ou=personnes,dc=societe,dc=com?mail?subtree?mail=*@gmail.com`

Protocole d'annuaire *LDAP*

- **Administrer OpenLDAP : *slapd.conf***
  - Le premier fichier de configuration d'OpenLDAP est "**/etc/ldap/sldapd.conf**" qui décrit les principaux paramètres de votre annuaire (voir **man slapd.conf** pour les nombreuses options). Voici un extrait de ce fichier:

```
# $OpenLDAP: pkg/ldap/servers/slapd/slapd.conf,v 1.8.8.6
2001/04/20 23:32:43 kurt Exp $
##
See slapd.conf(5) for details on configuration options.
# This file should NOT be world readable.
##
Inclusion des schémas nécessaires
include /etc/openldap/schema/core.schema
include /etc/openldap/schema/cosine.schema
include /etc/openldap/schema/inetorgperson.schema
include /etc/openldap/schema/nis.schema
# Options que vous pouvez modifier
#pidfile //var/run/slapd.pid
#argsfile //var/run/slapd.args
```

Bibliographie

- **Cours Administration réseau sous linux (Debian et Ubuntu)-*Rémy Malgouyres***
- **www.afnic.fr/ext/dns/**
- **http://www.linux-france.org/prj/edu/archinet/systeme - Cours sur les serveurs**
- **http://fr.wikibooks.org/wiki** - cours: Administration réseau sous Linux
- ***http://doc.ubuntu-fr.org/* - Documentation Ubuntu (serveurs)- Ubuntu-fr**
- **Guide Pédagogique OFFPT**
- **http://www.debian.org/doc/** - Documentation Debian (Serveurs)